BitTorrent Worm Sensor Network : P2P Worms Detection and Containment

Sinan Hatahet, Yacine Challal, Abdelmadjid Bouabdallah

BitTorrent Worm Sensor Network : P2P Worms Detection and Containment

Sinan Hatahet
Heudiasyc UMR 6599
Université de Technologie de Compiègne
BP 20529
60205 Compiègne cedex
France
Email sinan.hatahet@utc.fr

Yacine Challal
Heudiasyc UMR 6599
Université de Technologie de Compiègne
BP 20529
60205 Compiègne cedex
France
Email yacine.challal@utc.fr

Abdelmadjid Bouabdallah
Heudiasyc UMR 6599
Université de Technologie de Compiègne
BP 20529
60205 Compiègne cedex
France
Email boubdallah@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. In 2007, Worms have caused damages worth the amount of 8,391,800 USD in the United States alone. Nowadays, BitTorrent is becoming more and more popular, mainly due to its fair load distribution mechanism. Unfortunately, BitTorrent is particularly vulnerable to active worms. In this paper, we propose a novel worm detection system in BitTorrent and evaluate it. We show that our solution can detect various worm scans before 1% of the vulnerable hosts are infected in worst case scenarios. Our solution, the BitTorrent Worm Sensor Network, is built over a network of immunized agents, which their main job is to efficiently stop worm spread in BitTorrent.

INTRODUCTION

Peer-to-peer systems, like eMule, BitTorrent, Skype, and several other similar systems, have become 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 clients 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 communication 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 (CacheLogic, 2006).

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 [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]. All these features have made BitTorrent a leading P2P system in the Internet.

However, BitTorrent has a serious vulnerability, it utilizes a central server, known as a "tracker", to coordinate connections between peers in a "swarm", a term used to describe a BitTorrent ad-hoc file sharing network for a file (or a set of files) provided by a file distributor. The tracker of a swarm is specified by the swarm's original file distributor. The tracker doesn't implement selection mechanism on new arrivals and doesn't apply any control measures on the data declared by the system peers either. Moreover all peers in the swarm trust the tracker without implementing any authentication or verification procedures. Malicious nodes can take advantage of these facts and use them to their ends, and attack the infrastructure of BitTorrent. There are several attacks that can be conducted against BitTorrent. Such attacks are like Sybil attacks [6], Eclipse attacks [7], DDoS attacks, or even active worms attacks. Active worms are programs that self-propagate across the Internet by exploiting security flaws in widely-used services [8]. Given that P2P clients will unlikely be free of common exploitable vulnerabilities in the foreseeable future, an interesting research question is the feasibility of incorporating a self-defense infrastructure into a P2P network for the network itself to detect outbreaks of any unknown worm and contain its spread.

In this paper, we first present and analyze the spread of P2P worms that use various scan techniques. We then propose and evaluate our worm detection system, BitTorrent Worm Sensor Network (BWSN). We show that our solution can detect various worm scans before 1% of the vulnerable hosts are infected in the worst case scenario. The rest of the paper is organized as follows. In section 2, we first discuss how the BitTorrent works in practice and then we discuss the different worms' propagation in BitTorrent. We also present related and previous research done on P2P worms' detection and containment. In section 3, we explain the architecture of the distributed detection and containment system that we propose BitTorrent Worm Sensor Network (BWSN). In section 4, we present a model of BWSN. We then provide numerical analysis results of different detec-

tion scenarios, in section 5. We end this paper with conclusions and future work in section 6.

I.BACKGROUND

I.A. BitTorrent:

BitTorrent is a P2P protocol for content distribution and replication designed to quickly, efficiently and fairly replicate data [3]. BitTorrent is organized into 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 [10].

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*.

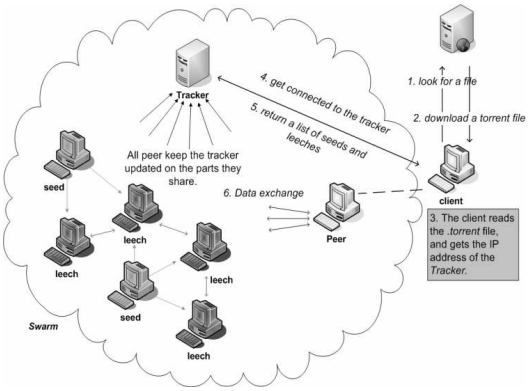

Figure 1: How BitTorrent Works

In figure 1, 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 1 step 1). The client would then download a *torrent* file which its metadata matches the desired file (see figure 1 step 2). Next, the client will read the content of the torrent file, and get the tracker address (see figure 1 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 1 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 1 step 5) [10]. 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, 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.

I.B. P2P Worms:

A computer worm is a program that propagates itself over a network, reproducing itself as it goes [11]. 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. Furthermore, there is a new trend of worms that is emerging and which has a huge destruction potential, such worms are called Peer-to-Peer worms. Based on the scanning strategies of P2P worms, they could be classified into two broad categories: passive worms and active worms. 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.

I.C. P2P Worms' propagation methods in BitTorrent:

We can classify P2P active worms into three categories: As a part of the Internet, P2P systems, like all overlay networks, could be attacked by *random scanning worms*. In this strategy, worm infected hosts do not have any prior vulnerability knowledge or active/ inactive information of other hosts. The worm-infected host randomly selects the IP addresses of victim targets from the global IP address space and launches the worm attack. When the new host is infected, it continuously attacks the system by using the same methodology. In this paper, this attack strategy is treated as the baseline attack for comparison purposes, as it has been widely adopted by many worms such as, Code-Red-I and Slammer. *Hitlist worms* attack a network using a pre-constructed list of potential vulnerable machines. 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. *Topologic worms* attack a network based on the topologic information found on their victims. The propagation of the Topologic worm has two phases: 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. Topologic worms choose their next victims in real-time. They employ 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 [11].

Unlike other P2P systems, BitTorrent doesn't allocate unique and long-life ID to its users. Consequently, Building a Hitlist consisted of BitTorrent nodes is very difficult if not impossible, as a result Hitlist worms tend to achieve a very low infection rate in BitTorrent. In the other hand, Topologic worms can achieve a much higher infection rate in BitTorrent, especially in crowded swarms where a single peer could be connected to 50 other peers.

II. RELATED WORK

Zhou et al. [12] propose a self-defense infrastructure inside a P2P network, by defining some independent "Guardian nodes" among the system peers. This approach is based on detecting worm's attacks on both the end hosts level, where those "Guardian nodes" have an automatic worm detection property, and on the network level where P2P overlays are treated like private networks. The role of a guardian node is to trigger an alarm when it detects an attack, in order to warn the other nodes of the system. Upon the reception of the alarm, the system nodes will take appropriate actions to become immune. The efficiency of this system depends on the placement of the guardian nodes in the P2P network topology and on the number of implanted guardians. To achieve optimal results with a minimum number of guardians, the authors suggest representing the P2P network overlay as a graph. Then, from this graph, a shortest path tree is built for each potential placement of a guardian. Using these trees as a decision criterion, the system's guardians are placed in a way to assure quick diffusion of any alarm.

Previous works such as [12] ignore BitTorrent architecture and use placement algorithms based on the graph theory, assuming that the majority of P2P systems have a topology that could be represented in graphs. This is, however, not the case of BitTorrent. BitTorrent has a complex architecture which is not node oriented; unlike most other P2P systems, BitTorrent has a file-oriented-topology. The BitTorrent architecture is composed of swarms, each swarm managing a unique file. A swarm as itself can be represented as a graph, owing to the nature of communication between its unique nodes. However, swarms are total independent bodies and no communication whatsoever exists between them. Representing BitTorrent as an activity graph where nodes are represented as vertices and intercommunication as edges is erroneous, especially that a node can join multiple swarms at the same time, and hence has multiple logical identities. In our work we focus on detecting active P2P worms very early in their life cycle through non-intrusive techniques. The main consideration is to limit any privacy concerns that could arise from conventional Internet monitoring through traffic sniffing, while still providing the fastest detection time possible. This has led us to investigate the efficiency of deploying a worm sensor network over the BitTorrent network (BWSN).

III. BITTORRENT WORM SENSOR NETWORK ARCHITECTURE

The BWSN is composed of interconnected and distributed agents over BitTorrent with the purpose of detecting worms breakthrough. Since P2P worms start their initial propagation by targeting only P2P hosts, common sense dictate us to deploy the BWSN agents as normal BitTorrent nodes. These nodes are immune and have automatic worm detection capabilities. Our main concern in the solution we propose, is to efficiently and quickly diffuse alerts to the system nodes, the worm detection techniques is out of scope of this paper.

A.BWSN Overview:

To better spread the alert of a potential worm attack, BWSN agents are auto organized into a topology. Once an agent detects a worm breakthrough it alerts the other agents, and since the agents are directly connected by a network the alerts could be diffused even faster. As soon as the agents are alarmed, each one will alert their respective trackers, which would alert the swarm's peers.

In figure 2, we illustrate how BWSN works. The *tracker* is installed on a machine which is located in a *swarm* repre-

sented by a cloud. Let's imagine a scenario, where a worm joins a swarm by contacting the tracker as a normal peer (see fig. 2 step 1). Once in the swarm, *leechs* will try to get connected to it, with the present agent among them (see fig. 2 step 2). Since an agent is designed like honeypots, it will succeed in getting connected to the worm quicker than the other *leechs* and eventually detect the attack (see fig. 2 step 3). Once the attack detected, the agent will immediately alert the other agents present in the swarm in order to share the workload of alarming the other agents (see fig. 2 step 4). Upon the reception of the alerts, agents will diffuse the alert through the BWSN (see fig. 2 step 5). While the alert spreads, the trackers will be in their turn alerted (see fig. 2 step 6). Finally, the trackers will alert their swarms' peers. The detailed algorithm is as follows:

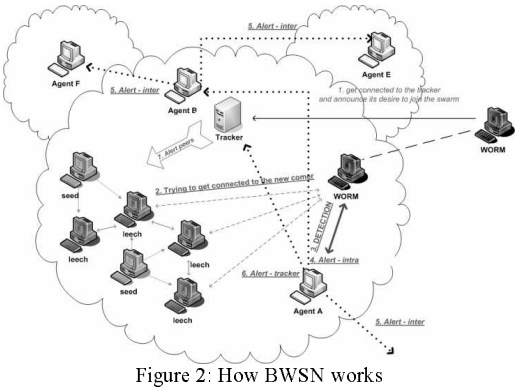

Figure 2: How BWSN works

Algorithm:

1. *a BWSN agent detects a worm attack at instant i.*
2. *the detecting agent alerts the other agents of the swarm*
3. *the alarmed agents participate in the propagation of the alert to the other non-notified agents (more details are given in subsection E).*
4. *each alarmed agent alerts the tracker managing the swarm(s) it is in, so the tracker alerts in its turn, the nodes it tracks.*
5. *Alerted nodes take an appropriate action to become immune to the worm attack (i.e. close an exploitable port, quit the swarm, install a patch, etc…)*

B. BWSN Agents Coordination:

BWSN provides a structure that allows collaboration between its agents. As soon as an agent detects a worm attack, it will trigger an alarm to the system. The purpose of alerts is for a recipient to learn enough information about the attack in order to take appropriate actions to become immune to the attack. With a small fraction of agents in each swarm, it is crucial that the system nodes are promptly alarmed. To assure a faster propagation of the alarm, the agent will first propagate the alarm to the other agents; upon the reception of the alarm the other agents will diffuse it within BWSN as well.

Alerts propagation through the BWSN is crucial to the well conduct of the system. In our approach, we chose to propagate the alerts using a Hitlist of agents. A Hitlist-based propagation will allow the agents to coordinate their efforts during the alarm diffusion and will guarantee a quicker propagation of the alerts too. All BWSN agents share the same Hitlist; it should be maintained and updated periodically by a centralized instance. During the alert propagation phase, when an agent is alarmed by a fellow agent, the alarming agent will ask him to alert half of its workload, and so on. This will allow the agent to propagate the alert in $\log_2 P / conn$ iterations, where P is the number of agents in BWSN and *conn* is the number of alerts an agent can diffuse simultaneously.

C. Increasing the probability of worm detection:

It is crucial that BWSN agents be able to detect a worm early, before the worm succeeds in achieving a high infection rate within BitTorrent vulnerable peers. To attain this end, the probability of BWSN agents being scanned by the attacking worm should be maximized. It is obvious that multiplying the number of agents in a swarm would maximize their chances of being scanned. As mentioned before, a node can join multiple swarms at the same time, and hence has multiple logical identities, and so can agents. So if we install n agents in the whole system and if each agent joins m swarms, we will have $n*m$ logical agents and therefore succeed in increasing the probability of agents being scanned m times. But the question remains: can we do better?

Actually we can, by exploiting the BitTorrent protocol. The main concern of the BitTorrent protocol is to quickly, efficiently and fairly replicate data. To optimally distribute data, each peer is faced with a number of important decisions to make, like which piece of the shared file to download and which peer it should serve (*i.e.* upload to). In order to do so, seeds choose to serve the leechs with the highest download speed, and leechs choose to serve the leechs with the highest upload speed. Connection speed is the key to be chosen and have an upper hand over the other peers of the swarm. Therefore, providing the BWSN agents with a high bandwidth would give them an advantage and would increase their probability of being scanned by the attacking worms.

D. Alert authentication:

Because alerts trigger actions at alerted nodes, an adversary could attack by disseminating bogus alerts. If alarms are issued directly from the agents to the rest of BitTorrent, there is no reason for a peer to trust the source, since agents (from BitTorrent peers view) do not have to be trusted especially that they behave like normal peers. To

avoid this problem, the responsibility of alarming BitTorrent nodes is attributed to the trackers, which are alerted directly by BWSN agents. Trackers are trust worthy authorities, that can verify the authenticity of alerts and they are trusted by the system peers as well. Moreover the trackers are the only entity in BitTorrent which have a global and detailed view of their respective swarm. After all, the job of trackers is to track the state of each peer in the swarm, hence the name. With this precious advantage, the tracker could alert its swarm peers before they get scanned by the worm.

V. MODELING AND PERFORMANCE OF BWSN

Before we evaluate our detection algorithm, first we need to understand how the number of victims increases during worm events. Then we need to set the parameters including the propagation time and the number of consecutive times that anomalies are observed. We choose these parameters based on observations and measurements conducted on BitTorrent [14]. In this section, we use traffic traces to decide the parameters and evaluate our detection algorithm.

V.A. Modeling the number of victims

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 appropriate and efficient detection and containment mechanisms.

Parameters: In order to formally build a model that describes the worm propagation, we need to indentify the different factors playing a role in its propagation. After a thorough examination and analysis we identified 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.

4) BWSN parameters: For the BWSN system, the following parameters need to be considered:

i) The number of alarms an agent can send at the same time.
ii) The number of swarms a BWSN agent can simultaneously join.
iii) The number of BWSN agents over BitTorrent.

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 a number of hosts are vulnerable, so our analysis considers the average case, we assume that each host 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 [14], that the size of a BitTorrent packet is 1 MB [13], and that the number of addresses returned by a tracker upon a request is 50 *peers* [10].

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

Parameters	***Notations***
S_i	Size of "P2P" system at instant *i*
P_0	Proportion of vulnerable hosts in the Internet
C	Attack capacity of worm infection host (number of victims being able to be scanned simultaneously)
$\lambda_{arrival}$	The rate at which a peer joins BitTorrent
λ_{leave}	The rate at which a peer leaves BitTorrent
λ_{comm}	The number of downloading request re-

	ceived 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 1MB)
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, BT)	The number of vulnerable hosts in P2P at the time *i* (*V*(0, *BT*) is the number of vulnerable hosts which can be infected at the system initial time = $S_0 * P_0$)
$V_p(i, BT)$	The number of vulnerable hosts in P2P and in protected swarms at the time *i*
I(i)	The number of infected hosts in the Internet at the time *i*
I(i, BT)	The number of infected peers in P2P at the time *i* (*I*(0, *BT*) is the number of initial infected hosts in the system.)
$I_p(i, BT)$	The number of infected peers in P2P and in protected swarms at the time *i*
newI(i, BT)	The number of newly infected hosts in P2P added at step *i* (*newI* (0,BT) = 0)
Conn	the number of alarms an agent can send at the same time.
App	The number of swarms a BWSN agent joins at the same time
numScan	The number of probes generated by the worm infected peers
CacheSize	The number of peers a peer can simultaneously upload to.
$N_j(i)$	The number of neighbors, *j* can attack at 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 ($avg_{leeches} = avg_{peers} *0.83$) [14]
Num	The number of peers in a swarm
$Pa(i)$	The number of alerted BWSN agents at instant *i*
P	The number of BWSN agents
$Tp(i-1)$	The number of alerted trackers in BitTorrent at instant *i*
T	The number of trackers in BitTorrent
λ_{Ti}	The proportion of alerted trackers in BitTorrent at instant *i*. $\lambda_{T(i-1)} = Tp(i-1)/T$

Table 1: Notations in this paper

Model: We adopt the epidemic dynamic model for disease propagation. In order to make it flexible for analyzing the various worms attacks, we use discrete time to conduct recursive analysis and approximate the worm propagation [9] [15]. In what follows, we will calculate the number of infected peers by the worms at instant *i* : *I(I, BT)*.

***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 decrements by the number of infected peers which left BitTorrent at the instant *i*.

Lemma 2: The number of probes generated by infected machines within BitTorrent is:

$$numScan = \sum_{j=1}^{I(i,BT)} \min[Cachesize, C] \quad (2)$$

Proof: Each infected host has neighbors as much as its cache can serve, therefore each worm-infected host can attack *Cachesize* P2P hosts. However, the number of scans a worm can simultaneously generate is limited by its capacity attack *C*, therefore a worm-infected host can generates as much as min[*Cachesize*, *C*] scans. In the other hand, the number of probes generated by infected machines during a random scanning worm attack is:

***Lemma 3*:** in BitTorrent, given the number of alerted BWSN agents *Pa(i)*, the number of scans generated by infected hosts *numScan (Lemma 2)*, and the size of BitTorrent *Si (Lemma 1)* at instant *i*, the number of alerted agents will be

$$Pa(i+1) = Pa(i) + [P - Pa(i)] * \begin{cases} \left[1 - \left(1 - \frac{\frac{P}{T} \times app}{S_i}\right)^{Numscan}\right] & (3)\ before\ worm\ detection \\ \left[1 - \left(1 - \frac{1}{P}\right)^{\sum_{j=1}^{Pa} N_{zona}}\right] & (3)\ after\ worm\ detection \end{cases}$$

Proof: The number of alarmed BWSN agents in BitTorrent before worm detection at instant *i* (4): is the number of previously alarmed agents in addition to the number of non-alarmed agents (*i.e.* $P - Pa(i)$) multiplied by the probability of being scanned by infected peers. The probability of an agent being scanned by an infected peer is bigger than the peer's, owing to the fact that an agent can join multiple swarms. Therefore, an agent's probability of being scanned is amplified by the number of its logical identities *app*, and P/T (*i.e.* the number of BWSN agents per swarm) since an alarmed agent immediately alerts the other agents of the swarm. Once an agent detects a worm, it spreads an alarm with the cooperation of its neighbors to the rest of agents in the BWSN. Therefore, the number of alerted agents *after*

worm detection at instant *i* (5): is the number of previously alarmed agent in addition to the number of non-alarmed agents (*i.e.* $P - Pa(i)$) multiplied by the probability of being alerted by alarmed agents. *Conn* is the number of alarms an agent can send at the same time.

Lemma 4: in BitTorrent, given the number of alerted BWSN agents *Pa(i) (Lemma 3)*, the number of alarmed trackers in the next tick will be:

$$Tp(i+1) = Tp(i) + \{[T - Tp(i)] * [1 - \left(1 - \frac{1}{T}\right)^{(\sum_{j=1}^{P-Pa(i)} conn) - (P - Pa(i))}]\} \quad (6)$$

Proof. The number of alarmed swarms (*i.e.* trackers) in BitTorrent at instant *i* (6): is the number of previously alarmed swarms in addition to the number of non-alarmed swarms (*i.e.* $T - Tp(i)$) multiplied by the probability of being alerted by alarmed agents. However the number of alerts that could reach the trackers is decreased by the alerts destined to the non-alarmed BWSN agents (*i.e.* $P - Pa(i)$) since alarming them is the priority of the system.

Proposition 1: in BitTorrent, given the proportion of alarmed trackers at instant *i* λ_{Ti} (*Lemma 4*), the number of alarmed vulnerable nodes (7), and the number of non-alarmed and vulnerable nodes (8) in the next tick will be

$$V_p(i+1, BT) = \lambda_{Ti+1} * (\lambda_{BTT} * V(i, BT) * 0.83 + V(i, BT) * 0.17 + \lambda_{aP2P}) \quad (7)$$

$$V(i+1, P2P) = [V(i, P2P) * (1 - \lambda_{lP2P} - \lambda_p + \lambda_{aP2P})] - V_p(i+1, P2P) \quad (8)$$

Proof. The number of alarmed vulnerable machines in BitTorrent at instant *i* (7): is the number of vulnerable leechs (*i.e.* $V(i, BT) * 0.83$) connecting to their respective trackers at instant *i* (*i.e.* multiplied by λ_{BTT}), in addition to the seeds of the swarms (*i.e.* $V(i, BT) * 0.17$) and the newly arrivals (*i.e.* λ_{aP2P}) multiplied by the proportion of alarmed trackers at instant *i* (λ_{Ti}= *Tp(i)/T, where Tp(i) is calculated in lemma 4)*. On the other hand, the number of vulnerable and non-alarmed peers in BitTorrent (8): is therefore, incremented by the number of newly arrived machines (calculated by the join rate λ_{aP2P}), and decremented by: *1)* the number of detected infected peers (calculated by the patching rate λ_p), *2)* the number of vulnerable peers which left BitTorrent at the instant *i* (calculated by the leave rate λ_{lP2P}), and *3)* the number of alerted vulnerable machines at instant *i*

Corollary 1: in BitTorrent, given the number of vulnerable peers *V(i,BT) (Proposition 1)*, the number of infected peers *I(i,BT)*, *the size of BitTorrent (Lemma 1)* and the number of number of scans generated by infected hosts *numScan (Lemma 2)*, the number of infected peers (9) in the next tick will be

$$I(i+1, P2P) = [I(i, P2P) * (1 - \lambda_{lP2P} - \lambda_d)] + newI(i+1, P2P) \quad (9)$$

Where,

$$newI(i+1, BT) = [V(i, BT) - I(i, BT)] * \left[1 - \left(1 - \frac{1}{S_i}\right)^{NumScan}\right]$$

Proof. The number of newly infected peers in BitTorrent *newI(i+1, BT)*: is the number of vulnerable but not infected peers (*i.e.* $[V(i, BT) - I(i, BT)]$) multiplied by the probability of being scanned by infected peers. Therefore, the number of infected peers in BitTorrent is incremented by the number of newly infected machine *newI(i+1, BT)*, and decremented by: *1)* the number of detected infected peers (calculated by the detection rate λ_d), and *2)* the number of infected peers which left BitTorrent at the instant *I* (calculated by the leave rate λ_{lP2P}).

V. EVALUATION OF BWSN

In this section, we evaluate the BWSN detection performance by using the above analytical model with different parameters for different scenarios. We report the performance results along with analysis.

V.A. Simulation Model:

Metrics: For each of the scenarios, we measure the attack detection system performance through calculating the infected hosts number (Y axis) over time (X axis). The higher the performance value, the worse is the detection system.

- *Parameters:* The general system is defined by the tuple: <*A, T, C,* S_0, P_0, λ_q, λ_c, λ_{BTT}, λ_p, λ_d, *I(0,BT), V(0,BT), Conn, num, App, CacheSize,* P, *T*>, representing the system configuration parameters. *A* determines the attack strategy and can be one of <*BTW, Topologic, Random scanning*>. Other parameters are presented in Table 1. As we are only focusing on selected important parameters that are sensitive to BWSN, the following parameters are set with constant values ($T=2^{32}$, *C=6, I(0,P2P) = 5,* λ_{BTT}*= 0.234, CacheSize =30,* S_0*=1,000,000, and V(0,BT) = 200,000)* in all our simulations.

V.B. Performance Results:

The preliminary results show that BWSN is a very efficient detection and containment system. Indeed, it allows reducing the infection damage with more than 99% in case of a topologic attack.

Fig. 3 shows the detection of a topologic worm attack. We notice that the worm was detected after 42 minutes and an

Supprimé : (

Supprimé : (9)

infection rate of 0,335%. It shows that BWSN is more efficient when the attacking worm targets only BitTorrent nodes at the beginning of its propagation, in other words BWSN is more efficient to detect topologic nature worms.

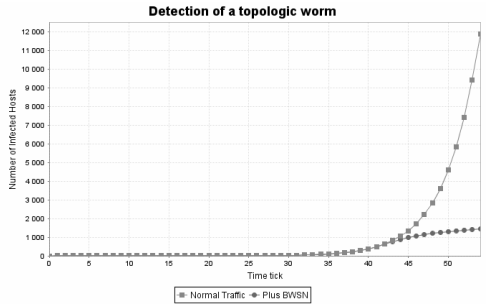

Figure 3

Fig. 4 shows the BWSN sensitivity to the number of swarms an agent can simultaneously join. We notice that the worm was detected after 42 minutes and an infection rate of 0,335% when an agent joins 5 swarms simultaneously. But when an agent joins 10 swarms simultaneously, the worm was detected after 41 minutes and an infection rate of 0,238%. In the other hand when it joins 15 swarms; the worm was detected after 39 minutes and an infection rate of 0,1195%. The results match our expectation the more swarms an agent can simultaneously join the quicker the worm is detected, due to the fact that an agent first alerts its colleagues that exist in the swarms it is currently in. This way when an agent joins multiple swarms, at the worm detection time it will have a higher number of agents to share its workload with.

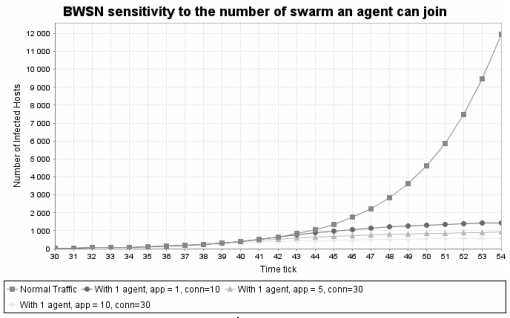

Figure 4

Fig. 5 shows the BWSN sensitivity to the number of alerts an agent can simultaneously generate. We notice that the worm was detected after 43 minutes and an infection rate of 0,445% when an agent joins 5 swarms and generates 10 alerts simultaneously. But when an agent joins 5 swarms and generates 30 alerts simultaneously, the worm was detected after 43 minutes and an infection rate of 0,335%. We notice that the agents don't achieve a notable change, however the infection rate evolution over the life cycle of the worm is less when the number of generated alerts per agent is higher.

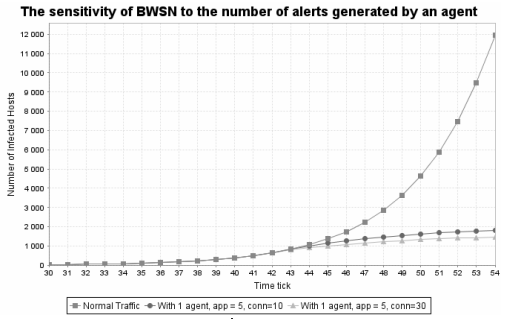

Figure 5

VI. CONCLUSION

The purpose of our research was to detect active P2P worms in BitTorrent and very early in their life cycle. Our main consideration was to limit any privacy concerns that could arise from conventional Internet monitoring through traffic sniffing, while still providing the fastest detection time possible. This has led us to investigate the efficiency of deploying a worm sensor network over the BitTorrent network (BWSN). In this paper, we proposed and evaluated our worm detection system BWSN. We showed that our solution can detect various worm scans before 1% of the vulnerable hosts are infected in the worst case scenario.

We built our system on two main bases. First, we built a network of detecting agents; this network allows the quick diffusion of a potential alert by supplying a mean of collaboration between its nodes (*i.e.* agents). Second, the collaboration between the system agents was implemented by sharing the workload of spreading the alert using a binary divide and conquer scheme. Future works could be focused on finding a more efficient workload distribution scheme and to imagine efficient containment scenarios and to implement them in BWSN.

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. 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.
11. Joukov, N. and Chiueh, T. Internet worms as internet-wide threat. *Experimental Computer Systems Lab, Tech. Rep. TR-143, September.* 2003.
12. *A First Look at Peer-to-Peer Worms: Threats and Defenses* . L. Zhou, L. Zhang, F. McSherry, N. Immorlica, M. Costa, S. Chien. 24-35, s.l. : Proceedings of Peer-to-Peer Systems IV, 4th International Workshop, February 2005
13. Bittorrent Protocol Specification v1.0. *Theory.org.* http://wiki.theory.org/BitTorrentSpecification.
14. Pouwelse, J.A., et al. The bittorrent p2p file-sharing system: Measurements and analysis. *International Workshop on Peer-to-Peer Systems (IPTPS).* 2005.
15. Chen, Z. S., Gao, L.X. et Kwiat, K. Modeling the Spread of. *In Proceedings of IEEE INFOCOM, San Francisco.* March 2003.