# A Distributed Stealthy Coordination Mechanism for Worm Synchronization

Gaurav Kataria, Gaurav Anand, Rudolph Araujo, Ramayya Krishnan and Adrian Perrig
Carnegie Mellon University

***Abstract*— Once a critical mass of nodes is infected by a worm it becomes very difficult to stop the worm from infecting a large fraction of vulnerable nodes. Therefore, the focus of strategies for worm defense has been to detect the worm before it reaches that critical mass. In this paper we present a novel distributed coordination technique for worm propagation and synchronization that can persist under the radar of detection mechanisms long enough to achieve critical mass for a full fledged attack. We discuss the stealthy worm propagation and synchronization approach exploiting a P2P file-sharing network.**

## I. Introduction

The emergence of flash worms in the wild, as hypothesized by Staniford et al. [24], [25], has stimulated interest in automated detection of Internet worms [9], [16], [23]. Early detection being crucial in immunizing hosts and/or setting up network filters, many techniques have been developed to detect and diffuse worms and viruses in early stages [14], [20], [26], [35]. In response, worm writers have attempted to develop even quicker propagation techniques to outpace these defensive mechanisms. Although faster propagation helps in gaining quick momentum, it also leads to early detection of worms, thereby helping the defensive mechanisms. Ma et al. have proposed self-stopping worms that stop scanning after a critical mass of nodes is infected in order to avoid further exposing infected hosts [10]. In this paper, we propose slow and covert propagation even in early stages of worm infection to avoid worm detection until a critical mass of nodes is infected. In addition, we propose a stealthy distributed coordination mechanism for worm synchronization that does not require command and control channels like IRC, which could be easily detected by network monitors. Instead, via use of discreet in-channel communication of an overlay network, a worm can spread and coordinate without raising an alarm.

P2P file-sharing networks can provide a vast overlay network ideal for stealthy propagation. P2P networks are not alone in providing an application-specific network overlay; email is another example, which has recently seen numerous worm outbreaks. Though what makes P2P file-sharing networks unique is that (1) they are large and distributed networks that are neither managed nor controlled by anyone, (2) due to enormous amount of data transfer that takes place on them, it is much easier to conceal malicious content and messages as part of regular communication, and (3) nodes connected to a P2P network by definition are aware of other nodes on the network and hence do not have to randomly scan the Internet for vulnerable hosts.

Interestingly, as technically savvy security researchers have been analyzing scanning-based self-propagating worms exploiting intricate software vulnerabilities; malware developers have been developing technically less savvy yet equally potent attack vectors using email and P2P file sharing. Of the more than twenty thousand worms and viruses reported by Symantec in year 2005, only a handful were scanning-based self-propagating worms that exploited a software vulnerability [27]. The majority of malware discovered was viruses, which is basically an executable code, compiled for a particular platform, typically MS Windows. The propagation vectors for these viruses are mostly email, P2P or instant messaging, instead of vulnerability scanning. It can be difficult to detect a virus if it does not exhibit any unusual behavior like opening a back door or altering the kernel. Therefore, in theory a P2P virus can persist under the radar as long as it behaves like a normal P2P application; surreptitiously using the P2P overlay network itself for distributed coordination with other infected nodes. In this paper, we present a stochastic model for P2P virus propagation and coordination such that once a critical mass of nodes get infected, all covert nodes become aware of that with high probability and can drop their cover to simultaneously launch a usual scanning-based Internet-wide worm infection. Effectively, we propose a novel malcode termed "Worus" that starts out as a virus and then morphs into a worm. A worus can potentially beat worm defenses that require a typical delay of $T_R$ after detection to activate filters over the global Internet (see Figure 1).

The remainder of the paper is organized as follows. In Section II we discuss the related work in the area of worm/virus propagation. Section III presents a background on P2P file sharing protocol. The Worus propagation model is discussed in Section IV, while Section V describes the distributed coordination scheme based on probabilistic counting. Sections VI and VII present a simulation model and its results. Section VIII proposes some countermeasures against the proposed Worus. Directions for future research and conclusions are presented in Sections IX and X, respectively.

## II. Related Work

Computer security researchers have applied the knowledge in the field of epidemiology to estimate the birth, death and cure rates of virus infections in computer networks to model the infection trajectory [15]. Zou et al. [36] developed a propagation model that fit the spread of the Code Red worm. Wang et al. [28] describe propagation models in terms of

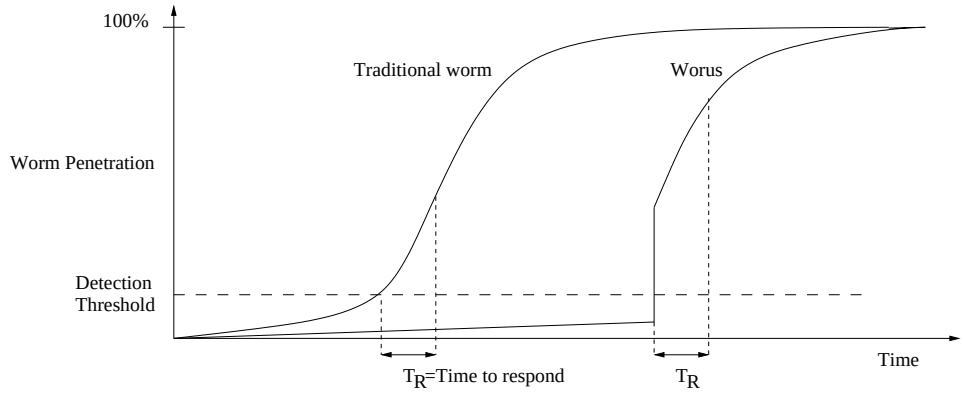


Fig. 1. This figure illustrates how a delayed yet quick ramp up of distributed stealthy worm coordination mechanism termed "worus" can gain higher penetration than a traditional worm, when defense techniques after detecting the worm activity exhibit a minimum delay of $T_R$ before successfully activating.

graphs, removing some of the constraints of previous models where homogeneous connectivity between nodes is assumed. Propagation models of other non-homogeneous systems, such as wireless networks [13] where the issues include dynamically changing connectivity levels, have also been developed. Zou et al. [34] modeled the spread of email viruses considering factors such as email checking time and the probability of a user clicking on an attachment. Similarly, instant messaging worms were analyzed by Mannan and van Oorschot [11].

On the other hand, considerable research has focused on developing techniques to specifically enhance (or contain) the severity of infection [24], [25], [29], [31]. Weaver et al. describe a number of new and highly virulent techniques that could be employed to increase infection rates and stealth [30]. The techniques proposed include hit-list scanning (where the worm contains a built-in list of vulnerable hosts) to create a Warhol worm; permutation scanning (where worms limit the wasted effort in scanning the same IP range multiple times) resulting in self-coordinated scanning; and flash worms which utilize Internet sized hit-lists. Our work is perhaps most closely related to what the authors describe as a surreptitious worm. However, during the later phase of our attack, the worm could be classified as a hybrid Warhol/flash worm.

While the security aspects of P2P have been studied in past, the focus has been on improving their architecture [5], [18], [19], [33], strengthening the P2P network, rather than considering the possibility that the P2P network itself may be used as an attack vector. Recently, some researchers have looked at this issue, for instance, Daswani et al. [6] simulate the impact of a Gnutella DoS caused using *query* flooding. McGann [12] describes how a self-replication attack can be launched through the Gnutella network and also discusses the susceptibility to Man-in-the-Middle attacks as well as port crawlers. Zeinalipour-Yazti [32] focuses on the exploitation of the inherent security weaknesses in the Gnutella protocol. The experiments they conducted outline the ease with which these security flaws can be converted into DDoS attacks, user privacy violations and IP harvesting mechanisms. They evaluate protection against such attacks and suggest certain improvements in the protocol. Bawa et al. [4] have developed a mechanism based on birthday paradox to determine aggregate content on a P2P network.

## III. Background

P2P file-sharing is a decentralized end-user application where millions of users across the world interconnect with each other on an ad hoc basis to share files. Although decentralized, P2P networks can be structured or unstructured. Popular P2P networks like Gnutella and KaZaA are classified as unstructured as the placement of data files is completely unrelated to the network topology. In contrast, on structured P2P networks such as Chord [5], CAN [18], PASTRY [19] and Tapestry [33], the nodes are positioned based on the content they offer. Content search in the latter is more efficient through the use of techniques such as distributed hash tables. On the other hand, searching on Gnutella is done rather randomly with queries being propagated along all paths except the path through which the query was received. This search continues up to a limit of 7 hops determined via a TTL field in the Gnutella packet header. In order to improve the efficiency of search in the Gnutella architecture, the Query Routing Protocol [1] was developed in which special nodes, called ultrapeers, maintain a hash table of content available at their respective leaf nodes. They perform a lookup within these tables before forwarding a query to the leaves in order to limit unnecessary query messages.

A typical host on a P2P network shares files by copying them to a specific folder and then publishing the contents of that folder to the network. The contents of the folder can then be searched by other users on the network and copied if desired by them. P2P viruses (or trojans) propagate by copying themselves to the user's shared directory under a deceptive name, such as a popular video game, song or a movie. The users who download these files also copy the virus and thus continue to propagate it on the file sharing network. In the past three years, over 300 such viruses have been found [2]. P2P viruses like other worms and viruses strive to steal either user information (for identity theft, fraud etc.) or computing resource (to form botnets, DDoS etc.).

## IV. Worus Propagation Model

The worus technique that we propose behaves like a normal virus in its initial phase when the malcode diffuses through user queries. The users query for files and receive responses, some of which are infected by the worus malcode. Once the user clicks on an infected file, its P2P client gets infected and starts returning worus infected files in response to queries from other users. However, at this time the worus does not commit any other malicious action(s) in order to avoid detection by early warning systems. Only after a critical mass of nodes get infected in this way, the worus transforms into a worm, starts to directly infect other hosts and performs its intended malicious actions.

A simplified model for the diffusion in virus phase can be formulated as follows:

$$N(t') = N(t) + \Delta(t) \tag{1}$$

Where,
$N(t)$: Number of infected nodes at time t; t'-t is one time unit
$\Delta(t)$: Number of nodes infected during time t'-t by the virus diffusion

The $\Delta(t)$ is calculated as follows:

$$\Delta(t) = p_i \times (N_0 - N(t)) \tag{2}$$

Where,
$p_i$ : Probability that an uninfected node is compromised within a time interval
$N_0$: Total number of nodes on the network

When a P2P user issues a query, the probability that his client gets infected depends on two factors. First, that at least one of the query responses received is infected and, second, that the user clicks on a infected response. We calculate the two probabilities below.

1. In order to determine the probability that a malicious *query-hit* makes it to the user's consideration set[1] we make the following intermediate computations,

Step 1: Experiments were conducted on Gnutella network to determine the average number of nodes that can be reached 0-6 hops away from any random node. The values observed based on whether the querying node was a leaf[2] or an ultrapeer are reported in Table I.

Step 2: With each node responding on an average to 1% of the queries they receive [8], we can determine the average number of responses that are received for any random query.

Step 3: A variable parameter $f$ is used to set the limit on the consideration set. Allowing this parameter to be a variable, ensures greater accuracy as this variable can be modeled based on usage patterns.[4]

Thus, depending on whether the querying node is a leaf or an ultrapeer, we can determine the probability that a query-hit in a user's consideration set is malicious, $p_m$.

2. For the sake of simplicity, we assume that the user is equally likely to select any response from the consideration set. Therefore, the probability of a query-hit being downloaded equals $1/f$. Hence, we evaluate the probability of getting infected,

$$p_i = p_m \times 1/f \tag{3}$$

In the following section we present the mechanism through which these infected nodes learn about and coordinate with other infected nodes on the network.

## V. Probabilistic Counting

In this section, we describe a novel probabilistic counting mechanism that the worus infected nodes use to monitor the state of infection on the network. Instead of an explicit communication channel, nodes rely on query and query-hit messages that are part of the P2P protocol to coordinate. Distributed and secret coordination between the nodes is an integral aspect of the attack we describe. It ensures that (1) every infected node launches the attack at the same time, and (2) malcode is not detected before it reaches the critical mass. Once the critical mass is achieved the worus immediately transforms into a worm and all infected nodes launch a synchronized attack. Synchronization is achieved through multiple stages of coordination described below.

Each node in the network, which is infected through the virus infection process described in previous section, can belong to one of three phases based on its knowledge of the spread of infection in the network. As the node learns of more infections on the network it progresses from phase 1 to phase 2 and so on. Therefore, as more nodes get infected, infected nodes progress to later phases. The purpose of having multiple phases is that with every successive phase change, the level of synchronization achieved across nodes is higher. This distributed synchronization is used to evade worm defenses by delaying the worm phase. We describe the phases and the synchronization process in detail in the next section.

### A. Phase Definitions

*1) Phase 1:* When a node is initially infected (by downloading a virus from a malicious peer), it is considered to be in phase 1. As the number of infected nodes on the network increases they need some mechanism to keep tract of their count. In order to estimate this count, each infected node generates **R** files with **b** bit long random filenames[5]. All

[1]The user's consideration set are those responses that are displayed on the screen to the user from which he chooses one.

[2]85% of nodes on a Gnutella network are leaf nodes [8].

[3]The anomaly in # of nodes reached for 5th and 6th hop for ultrapeer is attributed to the fact that several clients have implemented "high out-degree" wherein a client reduces the TTL of a query to 4 if it has a high out-degree in order to limit the query responses for optimizing bandwidth. In the case of a leaf node, the connection to its immediate ultrapeer is not counted as a hop.

[4]Casual discussions with many P2P users suggests a consideration set of 15-20.

[5]Generating random filenames as opposed to predefined filenames increases the stealth of the attack and reduces possibility of false positives.

TABLE I

REACH OF ULTRAPEER AND LEAF NODES IN A GNUTELLA NETWORK

| Hops | Number of nodes reached from ultrapeer | Number of nodes reached from leaf |
|---|---|---|
| 0 | 25 | 4 |
| 1 | 156 | 24 |
| 2 | 1867 | 1354 |
| 3 | 16465 | 12317 |
| 4 | 37429 | 32807 |
| 5 | 0[3] | 32807 |
| 6 | 0 | 7266 |

infected nodes then periodically[6] query for **r** ($\leq$ **R**) files each of which has a random filename of length **b** bits, expecting positive responses (query-hits) from other infected nodes on the network. A node continues to remain in phase 1 until it receives **t** ($\leq$ **r**) query-hits. On receiving **t** hits, the infected node progresses to phase 2. The same set of **r** files is queried periodically such that the phases are time-independent and depend only on the actual number of infected nodes in the network. We model the event of receiving a query hit as a Bernoulli trial and compute the probability of getting **t** hits using a binomial distribution. We use it to compute the probability of a node being in phase 1, i.e., getting less than **t** hits for the **r** queries sent.

With an estimate of the number of infected nodes in the network currently in phase 1, $N_1$, and the probability of being in phase 1, $Pr(hits < t)$, using recursion we can calculate the expected number of infected nodes in phase 1 as follows:

E[number of infected nodes in phase 1] = $N_1 \times Pr(hits < t)$

*2) Phase 2:* After receiving **t** hits for the **r** queries sent, an infected node in phase 1 progresses to phase 2. A node transitions into phase 2 when it becomes aware of a certain infection count having been reached in the network (based on the **t** hits received). In order to achieve synchronization among infected nodes, nodes in phase 2 have to speed up the process of informing other nodes in phase 1 of the current infection count. In order to do so, the nodes in phase 2 generate **R'-R** additional files so that nodes in phase 1 would have higher probability of receiving a hit. At the same time, they now query for **r'** files, which include the **r** files used in phase 1. The phase 2 nodes wait for at least **t'** ($\geq$ **t**) hits after which they can progress to phase 3.

With an estimate of the number of infected nodes in the network currently in phase 2, $N_2$, and the probability of being in phase 2, $Pr(queryhits < t')$, we can recursively calculate the expected number of infected nodes in phase 2.

*3) Phase 3:* Once a node gets into phase 3, it is ready to launch the attack because not only does it know that the infection threshold has been reached, but it also knows that a large proportion of other infected nodes also share the same knowledge. In order to speed up the process of getting the remaining nodes into phase 3 it generates **R"-R'-R** files. This is to help nodes that are still in phase 1 or phase 2 to quickly transition to phase 3. The expected number of nodes in phase 3 can be calculated as follows:

$$E[x''] = N(t) - E[x] - E[x'] - E[x'''] \tag{4}$$

Where,
$x$ = Number of infected nodes in phase 1
$x'$= Number of infected nodes in phase 2
$x''$= Number of infected nodes in phase 3
$x'''$ = Number of infected nodes in phase 4

*4) Phase 4:* When a node reaches phase 3, the worus turns into a worm and attempts to infect all the nodes, it can, using software vulnerabilities. The vulnerable nodes that get infected through this process are considered to be in phase 4. A node on entering this phase is ready to launch an attack (as in phase 3). The nodes in phase 4 also generate **R"** files to further speed up transition of nodes in phases 1 and 2, and also try to spread the worm to all hosts on and outside the P2P network. These two phases (phase 3 and phase 4) are essentially differentiated by the mechanism through which they are infected otherwise they exhibit the same behavior. As mentioned before, the goal of having multiple phases is to finally have an abrupt phase change, such that all nodes can attack simultaneously without any external trigger.

### *B. Analysis*

Probability of an infected node getting a query-hit is determined using birthday paradox [4],

$$p = \left(1 - e^{-(xR + x'R' + x''R'' + x'''R'')/2b}\right) \tag{5}$$

*1) Modeling Phase 1:* The probability of getting **t** hits from **r** queries is modeled as a binomial distribution,

$$P_t = C(r,t)p^t q^{r-t} \tag{6}$$

Where, q = 1-p, the probability of not getting a hit.

The probability of being in phase 1 is equal to the probability of getting less than **t** hits.

[6]The periodicity of these random queries has to be chosen appropriately to ensure a certain degree of synchronization.

$$Pr(hits < t) = \sum_{t=0}^{t-1} P_t \quad (7)$$

Hence,

$$E[x]_{new} = E[x]_{prev} \times P(hits < t) \quad (8)$$

Where,
$E[x]_{prev}$ := Expected number of nodes in phase 1 at the last infected count
$E[x]_{new}$ := Expected number of nodes in phase 1 at the current infected count.

*2) Modeling Phase 2:* The probability of getting **t'** hits from **r'** queries is,

$$P_{t'} = C(r', t')p^{t'}q^{r'-t'} \quad (9)$$

The probability of being in phase 2 is equal to the probability of getting fewer than **t'** hits.

$$Pr(hits < t') = \sum_{t=0}^{t'-1} P_{t'} \quad (10)$$

Hence,

$$E[x']_{new} = E[x']_{prev} \times P(hits < t') \quad (11)$$

Where,
$E[x']_{prev}$ := Expected number of nodes in phase 2 at the last infected count
$E[x']_{new}$ := Expected number of nodes in phase 2 at the current infected count.

*3) Modeling Phase 3:* From Section V-A.3 we have expected number of nodes in phase 3 as,

$$E[x'']_{new} = T - E[x]_{new} - E[x']_{new} - E[x''']_{new} \quad (12)$$

Where,
T := Total number of nodes currently infected
$E(x")_{new}$ := Expected number of nodes in phase 3 at the current infected count
$E(x''')_{new}$ := Expected number of nodes in phase 4 at the current infected count.

*4) Modeling Phase 4:* Number of infected nodes in phase 4 is modeled as explained in Section V-A.4.

## VI. Simulating a P2P network

The large scale interest in P2P systems has stimulated research efforts for simulating P2P networks. The GeorgiaTech network simulator [7] simulates a large-scale P2P network like Gnutella using a generic framework that supports packet level details. Similarly, a Query-Cycle simulator which attempts to accurately model real-life P2P networks is described in [22]. Our simulator, CMU-GNS, is built on top of a basic Gnutella simulator GnucNS developed by the makers of the Gnucleus Gnutella client [3]. The GnucNS simulator was designed as a tool to study the performance of the Gnutella network and has a number of programmable parameters. It supports version 0.6 of the Gnutella protocol and comes with support for features such as ultrapeers, node upgrade and downgrade algorithms, and average connectivity. CMU-GNS supports a number of extensions to the basic simulator. The network parameters have been adapted for bandwidth and average connectivity to more accurately reflect current statistics [8], [21]. Our simulator also more accurately follows the Gnutella RFC for the query generation and routing [1].

TABLE II
Simulation Parameters

| b | R | r | t | R' | r' | t' | R" |
|---|---|---|---|---|---|---|---|
| 17 | 15 | 10 | 6 | 85 | 15 | 11 | 500 |

The simulation takes place in two stages. During the first stage a lookup table is constructed that takes into account the current network topology and determines for a given number of malicious nodes how many of those are in each phase of the probabilistic counting mechanism. The second stage then uses the diffusion model developed above to increase the number of malicious nodes with time and at each time instant looks up the table generated during step 1, to determine which nodes would transition to the next phase.

## VII. Simulation Results

Using the simulator described above, we model the spread of infection on a small network of 500 nodes. The parameters were adjusted so that the final worm attack would be launched when the number of nodes ready to attack was close to 90%. The parameters used for the probabilistic counting are shown in Table II.

In Figure 2, we plot the results from the simulation, with the percentage of nodes in each phase against the total number of malicious nodes in the network. The number of non-malicious nodes in the network decreases steadily as infected nodes transition into higher phases. As expected, we observe a sudden transition of nodes into phase 3 which results in a synchronized attack. It is important to note that the actual time taken is dependent on the rate at which coordination queries are sent. There is indeed a trade-off here in achieving swiftness while maintaining low rate of queries to evade detection.

## VIII. Countermeasures

Gnutella network (much like other networks seen in nature) is scale-free and exhibits a heavy-tail or power-law nature. This means that a few nodes (usually the most popular and dependable ultrapeers) have a large outdegree while the remaining nodes have much lower outdegrees. This mathematically implies that the variance exhibited by the node degrees is infinite. Pastor-Satorras and Vespignani [17] show analytically and empirically that in a scale-free network the concept of an epidemic threshold, generally associated with epidemiology,

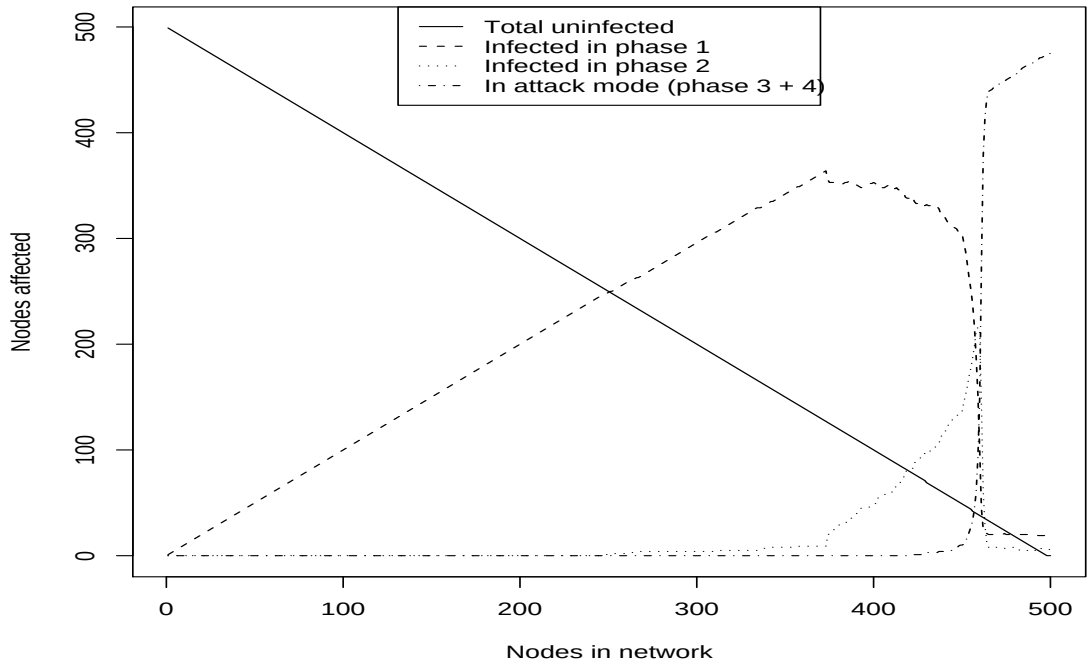


Fig. 2. This figure shows that initially almost all infected nodes are in phase 1, but as the number of infected nodes approaches 450 (i.e. =90% of nodes in the network), all infected nodes get quickly synchronized for a simultaneous attack.

does not exist and viruses with even low spreading rates can survive for long times. They also show that immunizing random nodes in the network has no effect. Therefore, none of the traditional immunization strategies would work as a countermeasure for the attack we described.

We have developed a countermeasure, which may not be able to completely stop the attack but can certainly reduce its severity by ensuring that fewer nodes participate in the synchronized attack. If used in combination with other worm detection and suppression techniques, it may render the attack ineffective.

We observed that the effectiveness of our attack stems from the fact that a large number of nodes simultaneously initiate the attack. Hence, if it was possible to artificially reduce the threshold at which a majority of nodes enter phase 3, and to also reduce the initial slope of the phase 3 curve, then, we could ensure that fewer nodes would participate in a synchronized attack, thereby limiting its efficacy. We therefore suggest that ultrapeers respond with bogus query-hits to querying nodes, in order to introduce noise in their counting mechanism and thus throw their synchronization off. Using the same parameters as those used to obtain the results described above, we implemented this countermeasure strategy. The results were encouraging, indicating earlier onset of phase 3 and a much slower transition. Figure 3 shows the results of the simulation with the countermeasures in use.

However, there are some limitations of such a defense. An innovative attacker might find it relatively easy to bypass this security mechanism by adjusting the tunable parameters that form the probabilistic counting method. By doing so, an attacker can make allowance for those query-hits which may have come from immunized ultrapeers. Therefore, we believe that a more advanced security scheme is warranted if such an attack is observed. Furthermore, the detrimental impact of implementing such a noise introduction scheme must be studied to observe any loss in functionality for normal users, who will be annoyed by seeing many bogus hits to their legitimate queries.

An effective approach, that has emerged in practice to filter out known P2P viruses is through use of rating systems for files on the networks. Instead of just relying on the filename, users are able to view other users' rating of the file as linked to its content hash, such that infected files are less likely to be hosted and copied by unsuspecting users. However, this approach is effective only against known viruses, and especially, against those viruses that have exhibited specific malicious actions. Moreover, considering the increasing sophistication of attacks we believe that it is likely that in future the virus (worus) infected clients could easily rack up the ratings of their viral files, enticing users to download them. Therefore, the attack that we have described could potentially become a serious threat that requires further research attention to find defenses.

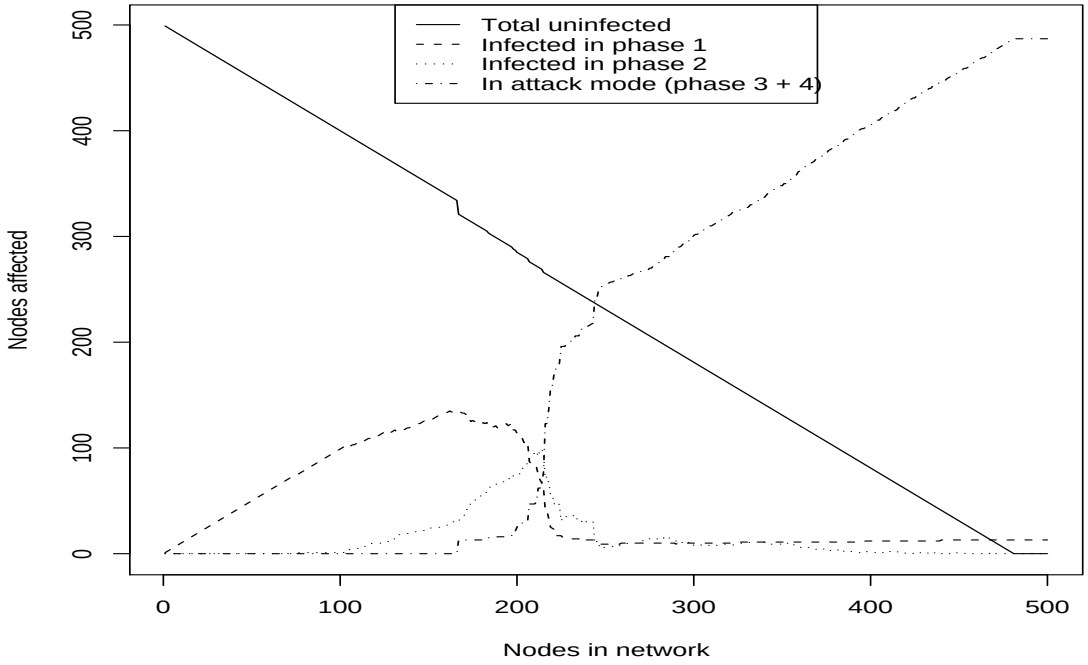


Fig. 3. This figure shows that by providing dummy hits the synchronization of infected nodes is thrown off. Infected nodes transition into phase 3 much sooner thereby reducing the efficacy of force multiplier effect.

## IX. Future Work

Though our simulator attempts to model the Gnutella v6 network accurately, its features could certainly be extended to support the dynamic nature of the network due to entry and exit of nodes as well as real-world Gnutella traffic. Network flux could play a significant role and potentially act as an inherent countermeasure because when malicious nodes go offline/online they can throw off synchronization. Certain parameters that are part of the diffusion model have been obtained through prior research work. Since most of the papers cited have based their results on experimental setups in Gnutella v4 networks, there exists a need to validate their findings on a Gnutella v6 setup. In addition, the assumption regarding equiprobable download pattern of query hits in a consideration set is simplistic and needs to be modeled more accurately through a proper user study.

As outlined earlier, the approach of rate limiting does not serve as a substantial countermeasure against the proposed attack and more research is required in this area. A possible extension could be to explore if the query pattern or in-channel communication of worus can be statistically detected in early phases, even if attackers generate plausibly random-looking files.

## X. Conclusions

Using simulation model of a P2P file-sharing network, we have shown that a covert yet formidable worm can exist. More importantly, the stealth during the initial phase, the quick ramp up to an attack state, and finally the full blown worm/DDoS attack only goes to show that serious thought must be put into the security of P2P networks, and overlay networks in general. We have successfully shown that combining two attack methodologies can only serve to increase the magnitude of the attack at the same time letting the attacker remain anonymous. The absence of a command and control channel differentiates Worus from other traditional worms. Time-bomb-based worms can also achieve similar synchronization in absence of command and control channel, but they may be set off too early in the worm propagation cycle or too late in the worm detection cycle.

When considering countermeasures, it is important to consider the impact they have on normal usage. Any improvement to the protocol that reduces the functionality or increases the possibility of being tracked is not likely to be acceptable to millions of file swappers on the Internet.

Finally, the worus approach proposed in this paper is general and can be applied to any popular overlay network like email or messaging. Therefore, more attention is required from the security research community to tackle these new threats before attackers implement them in the wild.

## Acknowledgment

The authors thank three anonymous reviewers for valuable comments. This research was supported in part by CyLab at the Carnegie Mellon University under grant CNS-0509004

from the National Science Foundation. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of Carnegie Mellon University, NSF, or the U.S. Government or any of its agencies.

## REFERENCES


[1] http://rfc-gnutella.sourceforge.net.

[2] http://www.symantec.com/avcenter.

[3] http://www.gnucleus.com/GnucNS/.

[4] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani, "Estimating aggregates on a peer-to-peer network," in *Technical report, Stanford University*, 2003.

[5] F. Dabek, E. Brunskill, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica, and H. Balakrishnan, "Building peer-to-peer systems with chord, a distributed lookup service," in *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, May 2001.

[6] N. Daswani and H. Garcia-Molina, "Query-flood dos attacks in gnutella," in *Proceedings of the 9th ACM conference on Computer and communications security*, 2002, pp. 181–192.

[7] Q. He, M. H. Ammar, G. F. Riley, H. Raj, and R. Fujimoto, "Mapping peer behavior to packet-level details: A framework for packet-level simulation of peer-to-peer systems," in *MASCOTS 03*, 2003, pp. 71–78.

[8] M. Kapadia and S. Bagla, "Peer-to-peer self-organizing communities," Master's thesis, Carnegie Mellon University, May 2003.

[9] C. Kreibich and J. Crowcroft, "Honeycomb - creating intrusion detection signatures using honeypots," in *Proceedings of the Second Workshop on Hot Topics in Networks (HotNets-II)*, November 2003.

[10] J. Ma, G. M. Voelker, and S. Savage, "Self-stopping worms," in *Proceedings of the ACM Workshop on Rapid Malcode (WORM)*, November 2005.

[11] M. Mannan and P. C. van Oorschot, "On instant messaging worms, analysis and countermeasures," in *Proceedings of the 2005 ACM Workshop on Rapid Malcode*, November 2005, pp. 2–11.

[12] S. McGann, "Self-replication using gnutella," in *Security Focus Online*. BugTraq Archive, May 2000.

[13] J. W. Mickens and B. D. Noble, "Modeling epidemic spreading in mobile environments," in *Proceedings of the 4th ACM Workshop on Wireless Security*, September 2005, pp. 77–86.

[14] D. Moore, "Network telescopes: Observing small or distant security events," in *Proceedings of the 11th Usenix Security Symposium*, August 2002.

[15] Y. Moreno, R. Pastor-Satorras, and A. Vespignani, "Epidemic outbreaks in complex heterogeneous networks," *European Physical Journal B*, vol. 26, no. 251, 2002.

[16] J. Newsome, B. Karp, and D. Song, "Polygraph: Automatic signature generation for polymorphic worms," in *Proceedings of the IEEE Security and Privacy Symposium*, May 2005.

[17] R. Pastor-Satorras and A. Vespignani, "Epidemic dynamics in finite size scale-free networks," *Physical Review*, vol. E65, no. 035108, pp. 1–4, 2002.

[18] S. Ratnasamy, P. Francis, M. Handley, and R. Karp, "A scalable content-addressable network (can)," in *Proceedings of ACM SIGCOMM*, 2001.

[19] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, November 2001, pp. 329–350.

[20] SANS, "Internet storm center. http://isc.incidents.org/."

[21] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "Measuring and analyzing the characteristics of napster and gnutella hosts," *Multimedia Systems*, vol. 9, no. 2, pp. 170–184, 2003.

[22] M. Schlosser, T. Condie, and S. Kamvar, "Simulating a file-sharing p2p network," in *1st Workshop on Semantics in Grid and P2P Networks*, 2003.

[23] S. Singh, C. Estan, G. Varghese, , and S. Savage, "Automated worm fingerprinting," in *Proceedings of the 6th ACM/USENIX Symposium on Operating System Design and Implementation (OSDI)*, December 2004.

[24] S. Staniford, D. Moore, V. Paxson, and N. Weaver, "The top speed of flash worms," in *Proceedings of the ACM Workshop on Rapid Malcode (WORM)*, October 2004.

[25] S. Staniford, V. Paxson, and N. Weaver, "How to 0wn the internet in your spare time," in *Proceedings of the 11th Usenix Security Symposium*, August 2002.

[26] Symantec, "Enterprise early warning solutions." [Online]. Available: http://enterprisesecurity.symantec.com/SecurityServices/content.cfm?ArticleID=1522

[27] D. Turner, S. Entwisle, O. Friedrichs, D. Ahmad, J. Blackbird, M. Fossi, D. Hanson, D. Cowings, D. Morss, B. Bradley, P. Szor, E. Chien, A. Burton, T. Conneff, P. Ferrie, T. Johnson, and D. McKinney, *Symantec Internet Security Threat Report, Volume VIII*. 20330 Stevens Creek Road, Cupertino CA: Symantec Corporation, 2005.

[28] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos, "Epidemic spreading in real networks: An eigenvalue viewpoint," in *22nd International Symposium on Reliable Distributed Systems (SRDS'03)*. IEEE Computer, October 2003.

[29] N. Weaver and V. Paxson, "A worst-case worm," in *Third Workshop on Economics and Information Security*, 2004.

[30] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, "A taxonomy of computer worms," in *First Workshop on Rapid Malcode (WORM)*, 2003.

[31] N. Weaver, S. Staniford, and V. Paxson, "Very fast containment of scanning worms," in *Proceedings of the 13th Usenix Security Symposium*, 2004, pp. 29–44.

[32] D. Zeinalipour-Yazti, "Exploiting the security weaknesses of the gnutella protocol," in *University of California Report*, May 2002.

[33] B. Y. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," in *UCB Tech. Report UCB/CSD-01-1141*, 2001.

[34] C. C. Zou, D. Towsley, and W. Gong, "Email worm modeling and defense," in *13th International Conference on Computer Communications and Networks*, October 2004, pp. 409–414.

[35] C. C. Zou, L. Gao, W. Gong, and D. Towsley, "Monitoring and early warning for internet worms," in *CCS03*. ACM, October 2003.

[36] C. C. Zou, W. Gong, and D. Towsley, "Code red worm propagation modeling and analysis," in *Proceedings of the 9th ACM conference on Computer and communications security*. ACM, November 2002, pp. 138 – 147.