INVITED PAPER *Special Section on Information and Communication System Security*

Concept, Characteristics and Defending Mechanism of Worms*

Yong TANG[†], **Jiaqing LUO**[††], **Bin XIAO**[††a)], ***and*** **Guiyi WEI**[†††], ***Nonmembers***

SUMMARY Worms are a common phenomenon in today's Internet and cause tens of billions of dollars in damages to businesses around the world each year. This article first presents various concepts related to worms, and then classifies the existing worms into four types– Internet worms, P2P worms, email worms and IM (Instant Messaging) worms, based on the space in which a worm finds a victim target. The Internet worm is the focus of this article. We identify the characteristics of Internet worms in terms of their target finding strategy, propagation method and anti-detection capability. Then, we explore state-of-the-art worm detection and worm containment schemes. This article also briefly presents the characteristics, defense methods and related research work of P2P worms, email worms and IM worms. Nowadays, defense against worms remains largely an open problem. In the end of this article, we outline some future directions on the worm research.

key words: *Internet worms, P2P worm, Email worm, IM worm, worm detection, worm containment*

1. Introduction

Worms are a common phenomenon in today's Internet and cause tens of billions of dollars in damages to businesses around the world each year. They compromise systems, steal sensitive information, remove files, slow down the network, and use the infected hosts to launch other kinds of attacks. Although a great deal of research has been done, defense against worm attacks remains largely an open problem and worm attack is still a huge threat to network communities for the following reasons: First, with the development of network applications, worms can exploit various ways to propagate themselves quickly. Some new worms appear recently, such as the P2P worm and IM (Instant Messaging) worm. Second, the spread speed of worms is much faster than human beings can manually respond. Third, worms are becoming more and more stealthy with the adoption of polymorphism and metamorphism techniques.

The paper is organized as follows. In Sect. 2, we will provide a clear description of various worm concepts. We classify the state-of-the-art worms into Internet worms, P2P worms, Email worms and IM worms. We respectively explain the characteristic and defending mechanism for each worm type (from Sect. 3 to Sect. 6) where we focus on the survey of Internet worms. Finally, we outline some future worm research directions in Sect. 7.

Manuscript received December 1, 2008.
Manuscript revised March 4, 2009.

[†]The author is with the College of Computer, National University of Defense Technology, China.
[††]The authors are with the Department of Computing, Hong Kong Polytechnic University, Hong Kong.
[†††]The author is with the College of Computer Science and Information Engineering, Zhejiang Gongshang Univ., China.

*The work was partially supported by the National Basic Research Program of China (973) under Grant No. 2005CB321801, and NSFC under Grant No. 60673167 and 60803161.

a) E-mail: csbxiao@comp.polyu.edu.hk

DOI: 10.1587/transinf.E92.D.799

2. Worm Concept and Category

2.1 Concept of a Worm

We use a broad definition of a worm proposed by Kienzle and Elder [1]– "**A worm** (also refers to **computer worm** or **network worm**) is malicious code (standalone or file-infecting) that propagates over a network, with or without human assistance". Note that in this article we do not distinguish between computer worm and network worm.

Compared with worm, malware is a more general concept. A malware is a software designed to infiltrate or damage a computer system without the owner's informed consent. Malwares include viruses, worms, trojan horses, bots, spyware, dishonest adware, most rootkits, and other malicious and unwanted software. Various malwares are defined mainly based on their functions and the lines between different malwares are not all sharp. Worms differentiate other malware classes, such as viruses, mainly in that worms actively use network interfaces for propagation. Nowadays, more and more worms act the role of a carrier for other malwares. As Fig. 1 illustrates, many malwares, like trojan horses and bots, can be loaded behind worms. For example, some variants of Agobot worms are combinations of worm, backdoor and bot. They are worms since they can propagate in Internet, they are backdoors since attackers can bypass security mechanisms and access computer resource, they are bots since they have command-and-control infrastructures.

2.2 Worm Category

After studying a great number of worms and comparing existing definitions of various worms, we classify worms into the following four classes, *according to the space in which a worm finds the target*.

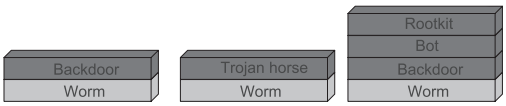

Fig. 1 Malwares behind worms.

- **Internet worms:** Internet worms are worms that find the target in the IP address space. These worms self-propagate in the Internet by exploiting security flaws in computers.
- **P2P Worms:** P2P worms are worms that find the target in the space of P2P networks.
- **Email Worms:** Email worms are worms that find the target in the space of email addresses. Email worms self-propagate by sending infected email messages.
- **Instant Messaging (IM) Worms:** IM worms are worms that find the target in the space of IM user IDs.

It is important to note that the above classification is not strict. A number of worms appear in two or more classes. For example, the Nimda worm is an Internet worm, also an email worm. Bibrog is a P2P worm and email worm.

2.3 Human Intervention Degree of Worms

A worm's propagation typically includes the following three phases: target finding, worm transferring, and infection. In the first phase, worms decide the next victim in different spaces, such as IP address space. After finding a target, worms transfer themselves to targets in the worm transferring phase. In the infection phase, the worms' codes will be executed, which have been transferred in the worm transferring phase.

In the first phase, we use the target finding space as the criterion to classify worms, as introduced in Sect. 2.2. Worm transfer and infection in the later two phases can be done manually or automatically. From high to low, there are four human intervention degrees: manual transfer & manual infection, manual transfer & automatic infection, automatic transfer & manual infection, and automatic transfer & automatic infection.

Figure 2 shows the human intervention degrees of four worm classes. Internet worms usually perform automatic transferring and automatic infection via remotely exploitation in which worm codes can be transferred and executed without any human actions. IM and Email worms perform automatic transferring & manual infection (e.g., sending worm codes by email or IM software, but these codes require human action to be executed), or manual transferring & automatic infection (e.g. after users visit fishing Web sites, a manual process, malicious codes will be executed automatically due to the JPEG vulnerability [2], a automatic process). P2P worms have various propagation methods covering all four human intervention degrees.

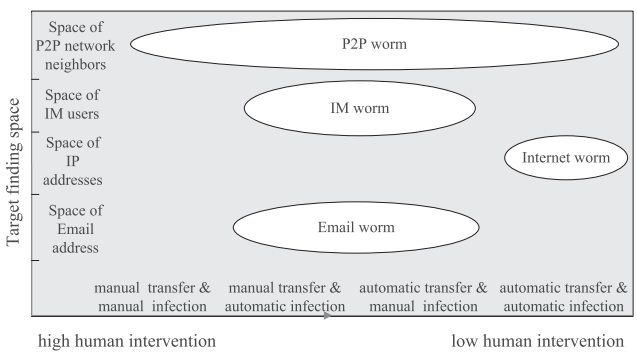

Fig. 2 Four worm classes and their human intervention degrees.

3. Internet Worms

3.1 Characteristics of Internet Worms

We describe the characteristics of Internet worms based on three factors: target finding strategy, propagation method, anti-detection technique, as Fig. 3 illustrates. The target finding strategy represents how an Internet worm finds new targets to infect. Propagation method represents how an Internet worm infects and copies itself to new targets. Anti-detection techniques are used by worms to evade detection.

3.1.1 Target Finding Strategy

Before an Internet worm infects a machine, it must firstly find the targets in the IP address space. There are a number of target finding strategies which can be classified three main classes: blind, passive and hit-list. Note that worms could use more than one strategies to discover new targets.

Blind: Blind scanning is one simple way, where worms blindly scan the entire Internet IPv4/IPv6 address space without prior knowledge about the targets. There are four types of blind scanning:

- *uniform scanning*: worms (e.g., Code Red and Slammer worms) uniformly and randomly pick IP addresses in the entire IP address space as the targets. Uniform scanning is the simplest scanning strategy.
- *sequential scanning*: worms (e.g., Blaster worms) sequentially scan IP addresses after randomly selecting a starting IP.
- *local preference scanning*: worms (e.g., Code Red II worms) prefer to scan the IP addresses within the

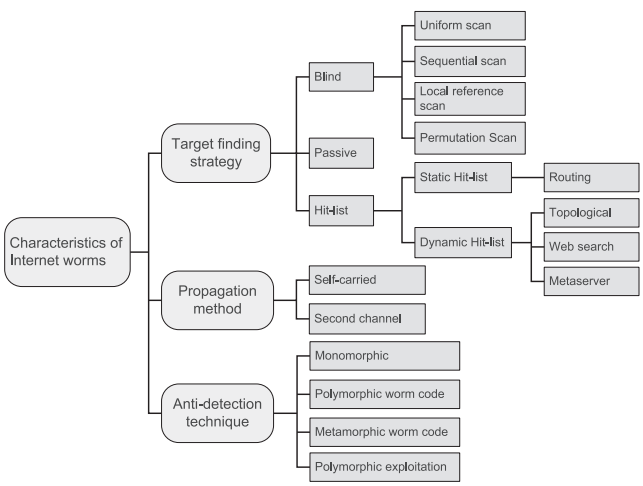

Fig. 3 Characteristics of Internet worms.

same subnetwork. Local preference scanning worm propagates faster than uniform and sequential scanning worms, since Internet IP space is not uniformly allocated.

- *permutation scanning*: all worms permutate the IP address space into a common virtual one and launch sequential scanning in this virtual IP address space with different starting IP addresses. Permutation-scanning worms propagate much faster, since they minimize the duplication of scanning. Warhol worm [3] is a worm very close to permutation-scanning worm.

In general, blind scanning worms may be easier to implement, but they have the following disadvantages: first, the miss rate of IP scanning can be very high, which can be used for worm detection. Second, they are not so fast, compared to other target finding strategies. Third, they are not efficient in the networks where IPv6 or Network Address Translation (NAT) is used. According to the simulation by Zou et al. [4], a worm will take 40 years to infect half of the vulnerable hosts in a single/64 IPv6 network.

Passive: A passive worm (e.g., CRClean worm [5]) does not aggressively seek the target by scanning. Instead, it passively waits for potential victims to contact the machine where the worm resides, and then tries to infect the visitors during the interaction with them. Although they are very slow, the worms using passive target finding strategy will be very hard to detect since they do not produce any anomalous traffic during target finding.

Hit-list: A hit-list is a list of known addresses as target hosts, which must be identified before the actual attack. A hit-list scanning worm uses a hit-list to accelerate the propagation speed. The strengths of hit-list is that worms can propagate much faster and stealthier, since the connection failure rate is relatively low. The hit-list for one worm instance can be created statically or dynamically:

- *Static hit-list* is created before a worm is released and a same hit-list is carried in each worm instance. For example, a *routing worm* carries a snapshot of BGP routing tables as the static hit-list. Routing worms not only propagate very fast (about three times faster than traditional worms), but also can conduct pinpoint "selective attacks" (e.g. attacking specific country) by using the geographic information of BGP routing prefixes [4]. The weakness of static hit-list scanning is that a hit-list could be out of date and the bigger size of the hit list, the harder it is to carry for a worm.
- *Dynamical hit-list* is dynamically created in each infected machine. There are several methods to create dynamical hit-list. *Topological worms* search local communication topology information (e.g., the hosts stored in the /etc/hosts file) in the infected machine to find new targets. *Web Search worms* [6] use search engines such as Google to find vulnerable targets. Some network services maintain lists of available servers/hosts (e.g., the Gamespy service maintains a list of servers for several games), which could be used by *metaserver worms* [5].

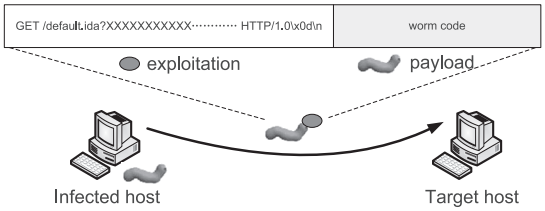

Fig. 4 Self-carried propagation of a worm.

3.1.2 Propagation Method

Most Internet worms propagate by exploiting the vulnerability(-ies) of softwares. If we focus on how a worm transfers itself to the victim, there are two propagation methods– *self-carried* and *second channel* [5]:

Self-carried: A worm infection attempt consists of an exploit and a payload, where the payload is the malicious codes embedded or appended to the exploit. For a self-carried worm, the payload is straightforwardly the worm code or the encoded worm code. Figure 4 illustrates the infection attempt of the Code Red II worm, a typical self-carried worm.

Second channel: A number of worms transfer themselves through a secondary communication channel. In other words, these worms only transfer a small piece of malicious code (such as a shell code) to the target during the infection process; then the worms' full codes are downloaded/updated from the infecting machines, public servers, or through a botnet [7].

3.1.3 Anti-Detection Technique

Worms could use various kinds of anti-detection techniques to evade detection systems.

Monomorphic: Monomorphic worms do not change their worm code and always send the same infection attempt to the targets. Monomorphic worms can be easily detected via signatures.

Polymorphic worm code: A worm polymorphic in code changes its binary code by encryption while keeping the original worm code intact. While such a worm can take millions of different forms by mutating its encryptions, the decrypted worm body is unchanged. Hence, we can get the decrypted code and then detect it by using a code emulator [8], [9]. TAPiON is an example polymorphic tool and Agobot is an example worm polymorphic in code.

Metamorphic worm code: Metamorphic worms in code are able to create new generation of worms in which the code is transformed or changed. An ideal metamorphic worm can not be detected by any signature. Some published metamorphism techniques include permutation of subroutines, insertion of garbage/jump instructions, and code substitution. ADMmutate, Clet are two well-known metamorphism tools. Writing a perfect metamorphic worm that can

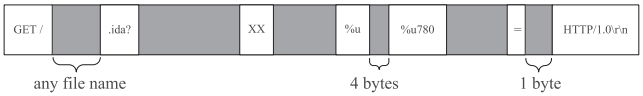

Fig. 5 Polymorphic exploit based on Code Red II worm. Shaded content represents wildcard bytes. Unshaded content represents invariant bytes.

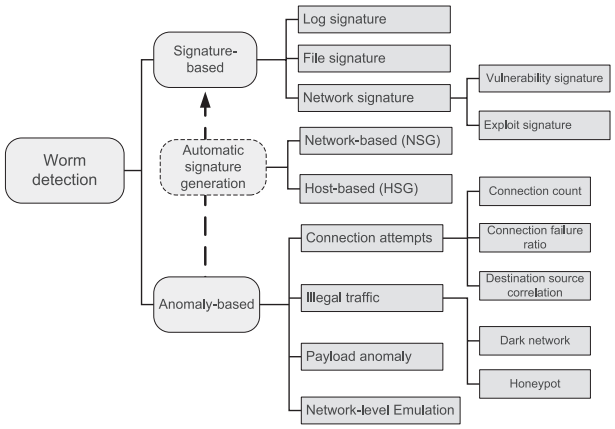

Fig. 6 Worm detection schemes.

intelligently change its behaviors or even evolve itself is attractive for worm writers.

Polymorphic exploitation: A worm infection attempt consists of exploit and payload. Payload can be dynamically changed through polymorphic or metamorphic worm code. Exploit can be made polymorphic by mutating some unimportant bytes (these are called wildcard bytes), but keeping some bytes intact (these critical for a successful infection are called invariant bytes). Figure 5 shows an example polymorphic exploit adapted from the un-polymorphic exploit of Code Red II worm in Fig. 4.

Usually, if a worm uses either the techniques of polymorphic/metamorphic worm code or the techniques of polymorphic exploitation, we call it a polymorphic worm. A perfect polymorphic worm does not contain any artifact (e.g., invariant bytes) and hence can not be detected by signature-based systems. Fortunately, most recent worms (e.g., Nimda, Slammer, Sassar and Witty worms) are still monomorphic worms and no perfect polymorphic worm has been reported. But any way, polymorphic worm is becoming an imminent threat for Internet.

3.2 Internet Worm Defense

The defense of Internet worm involves two steps: worm detection and worm containment. Worm detection is to find the activities of Internet worms. As Fig. 6 shows, the current worm detection techniques are roughly classified into signature-based (Sect. 3.2.1) and anomaly-based schemes (Sect. 3.2.2). Automatic signature generation (Sect. 3.2.3) is a new technique can connect these two detection schemes. Worm containment (Sect. 3.2.4) is to react quickly and minimize the damage after a worm is detected.

3.2.1 Signature-Based Worm Detection

Signature-based detection is a traditional approach widely used in intrusion detection systems (IDSs). In signature-based detection, the patterns or the behaviors of the worms are modeled; once a match is detected, the detection system will be raised. There are different signature types: *Network signature*, such as regular expression, which aims to match every infection attempt (network flow) of a worm. *Log signature* existed in application and system logs can expose the behaviors of a worms in the victim host. *File signature* represents the tracks of worms in the file system. In this article, we focus on network signature, since log signature and file signature traditionally belong to the research scope of virus detection. In nature, there are two subtypes of network signature: exploit signature and vulnerability signature.

Exploit-based signature: An exploit-based signature describes the characteristics of one or a few exploitation(s). A good exploit-based signature can detect the worm infections that make use of the same exploit, even the infections are polymorphic. Although exploit-based signatures can only detect known exploits/worms, so far, they are still irreplaceable, especially for the early and timely detection of zero-day worms, which utilize zero-day exploits[†].

Vulnerability-based signature: A vulnerability-based signature describes the properties of a certain vulnerability and hence can detect all possible exploits utilizing this vulnerability. Vulnerability-based signatures are more effective than exploit-based signatures in that they can detect unknown exploits if a vulnerability can be utilized by many exploits. However, generating vulnerability-based signature is very complicated and time consuming [10].

Most IDS/anti-virus vendors provide both types of signatures, telling us both what vulnerability has been exploited and what types of exploits are used for a worm.

3.2.2 Anomaly-Based Worm Detection

Anomaly-based detection systems build the models of normal network or program behavior; when the behaviors of a host or program departs from these models, a alarm will be generated. As seen in Fig. 6, anomaly-based detection methods are roughly based on connection attempts, illegal traffic and payload anomaly [11].

Connection attempts: In order to propagate faster, worms send a large number of TCP SYN packets or UDP packets to find victims in a very short period. Hence, we can detect worms using the following straightforward strategies:

- *Connection attempt count*: If the number of SYN packets sent from a certain host exceeds a threshold value within a period of time, the host is considered to be infected.

[†]A zero-day exploit is an exploit that take advantage of unknown, undisclosed or unpatched vulnerabilities.

- *Connection failure count/rate*: In a short period of time, if a certain host receives a large number of connection failure packets (i.e., TCP RST packets, or ICMP host unreachable packers), or the ratio of success and failure connections is high, the host is considered to be infected. This method is useful for blind scanning worm, but less effective against hit-list, topological and passive scanning worm.
- *Destination-Source correlation*: Worms fast move in the vulnerable hosts. That is, if a host is infected in a way (e.g., the packets designated to a certain port or contain similar content), soon, he will try to infect other host in the same way. Hence correlating the input packets and the output packets can discover the movement of worms [12].

Illegal traffic: There are a large portion of addresses unused in the Internet. Normal machines rarely send packet to these unused address, however worms do. If illegal traffic means the network traffic to unused addresses, we can detect worms based on illegal traffic.

- *Darknet*: A darknet (also called black hole network) is a portion of routed, allocated IP space in which no active services or servers reside. These addresses are "dark" because there is, seemingly, nothing within these networks. Since any traffic toward darknet are likely from worms. Team Cymru maintains a darknet project [13], Vinod Yegneswaran et. al. describe a darknet, the Internet Sink (iSink) system, which measures packet traffic on unused IP addresses in an efficient, extensible and scalable fashion [14].
- *Honeypot*: A honeypot is a vulnerable system on the network that does not provide any real/product services. Similar to darknet, any traffic to honeypot is suspicious. The difference is that honeypot responds to visitors, however darknet does not. As Fig. 7 shows, there are three honeypot implementation types – physical honeypot (responds through physical machine and real system) [15], emulated honeypot (responds through emulated OS and service) [16]–[18], and virtual honeypot (responds through real OS and network services running in a virtual machine) [19]. Since honeypots can communicate with visitors in a higher interaction, they can accurately detect all kinds of worms and then capture the worm samples [20].

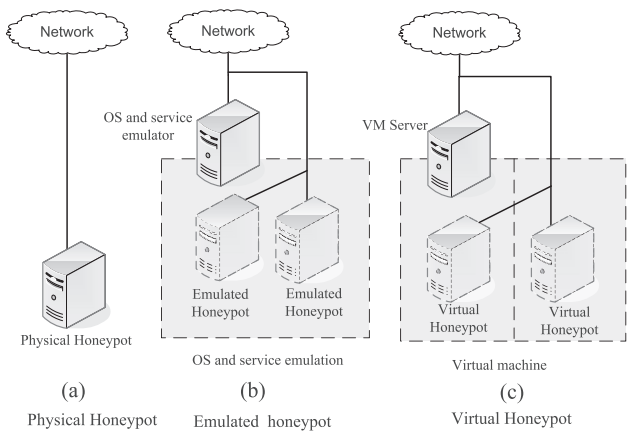

Fig. 7 Three types of honeypot implementation.

Payload anomaly: Payload anomaly detection first builds the models of legitimate payloads, and then detects anomalous network activities by check whether the payload of a packet/stream satisfies the normal models. PHAD [21] is a system builds the models of packet head fields, Anagram [22] and PAYL [23] are the system build the models of the legitimate packet content. In general, building a solid model of all possible legitimate payload is difficult because of the diversity of network application. But, it is possible for a specific service or network site. Christopher Kruegel et. al. proposed such a system [24]. Note that payload anomaly detection approaches are challenged since worms may embed deliberate legitimate traffic in the payload to evade detection.

Network-level Emulation: The payload in worm infection attempts (as seen in Fig. 4) likes a program that could be executed in the victim host. Network-level emulation has recently been proposed as a method to accurately detect these program-like payloads from polymorphic worms, based on the actual and dynamic execution of network data on an instruction emulator. Such systems include [8] and [9]. Their experiments demonstrate high detection rate and near zero false positive rate for current polymorphic exploits.

3.2.3 Automatic Signature Generation

In general, signature-based detection systems are accurate, efficient, and easy to develop and deploy. However they can not detect unknown worms unless new signatures are available. On the contrary, anomaly-based detection systems are able to detect unknown worms but they usually suffer from high false alarms (false positives) since modeling normal behaviors is very difficult. Fortunately, automatic signature generation, an emerging technique, can be a bridge to combine the advantages of signature-based detection and anomaly-based detection– We can use anomaly-based detection systems to discover unknown attacks (worms), and then use automatic signature generation technique to generate accurate signatures for detection. In recent years, automatic signature generation has been an active subject and a number of systems have been proposed. In terms of input information, these systems can be broadly classified as either host-based or network-based.

Host-based signature generation (HSG) HSG systems run in the protecting hosts and make use of host information to detect the attempts of infection and generate signatures from these attempts. Typical HSG systems include [25]–[28]. In general, host-based approaches generate accurate signatures quite quickly but usually have negative effects on the protecting host in terms of performance and

Table 1 Scope and location of worm defense.

scope	location	advantage	disadvantage
Host level	located at host	• accurate in known worms detection, since many host information can be used	• only one infected host could be contained for one deployment. Thus, blocking the propagate of a worm in the Internet needs very high deployment rate.
LAN level	located at border router, the gateway of LAN or sub-LAN	• flexible strategies for detection, such as Honeypot, can be used; • a good place to contain worm infections, since all traffic going in and out of the network can be inspected	• may have the single point of failure problem
Global level	located at backbone routers	• more effective to contain worm in global scope, because not all LANs deploy worm defense systems	• not very practical: it requires wide cooperation among ISPs, organizations, and countries. However, few organizations are willing to share their data with others.

configuration, e.g., the host need to recompile the kernel or modify runtime libraries.

Network-based signature generation (NSG) NSG systems solely analyze the suspicious network traffic and output content-based signatures. Comparing with HSG systems, NSG systems are more sensitive at the early stage of worm propagation, since they can capture worm samples earlier from the benefit of working at the network router/gateway level. Early network-based signature generation approaches including Honeycomb [20], PAYL [23], and EarlyBird [12] were able to generate only single-string signatures. Polygraph [20] and Hamsa [29] are two token-based approaches that try to select a set of tokens that have high coverage of the suspicious pool and low false positive reactions to the normal traffic pool. SRE [30] uses multiple sequence alignment to generate signatures.

3.3 Worm Containment

After a worm is detected, reacting quickly and minimizing the damage is very important. That is why worm containment is required and why we say that worm defense = worm detection + worm containment. Worm containment aims to limit a worm's propagation by isolating it in a small subsection of the network. Basically, there are three worm containment methods [11]: (1) *Slowing down*: When a host is identified as worm victim, any traffic from this host address will be slowed down. (2) *Address blocking*: When a host is identified as victim, any traffic from this host will be dropped. Address blocking must be very careful, since in the case of false positives, noninfected hosts might get blocked off. (3) *Content blocking*: If packet content matches a worm signature, the packet will be dropped. Content blocking has been widely used by security products, such as firewall, IPS (Intrusion Protection System), and IDS.

3.4 Scope and Location of Worm Defense

Worm defense can be achieved in different locations with different scopes. From low to high, there are three worm defense levels:

Host Level: Defense systems in this level, such as host-based firewalls, HIDSs (Host-based intrusion detection systems), anti-virus softwares, are deployed at a host and monitor all kinds of information in the protected host.

LAN Level: LAN level worm defense systems, such as Firewall, NIDS, IPS, and Honeypot etc., are located at border router, the gateway of LAN or sub-LAN. Defense in LAN level is considered the most effective worm defense way for enterprise networks because of its good detection accuracy and containment ability.

Global Level: Global level worm defense works in an ISP or AS, or even the whole Internet, usually located at backbone routers. Several global level worm defense methods have been proposed, such as DAW (Distributed Anti-worm Architecture) system [31] and IMS (Internet Motion Sensor) system [32].

The comparisons in terms of advantages and disadvantages of the three worm defense level are provided in Table 1. In general, as the scope grows bigger, detection may be rougher, but containment is more effective [11]. Nowadays, host level systems have been deployed most widely, LAN level systems are mainly deployed in enterprise networks, global level systems still stay in the research stage.

4. P2P Worms

P2P worms neither need to probe random IP addresses nor generate high rates of failed connections. As a result, they could cause less abnormal behaviors which lead to more stealthy in detection. In recent years, P2P worms have already emerged in P2P file sharing systems like BearShare, KaZaa and Gnutella, etc.

4.1 Target Finding Strategies of P2P Worms

Usually, P2P worm was viewed as a form of topological worm which searches local information to find next victims. But, the high penetration of P2P protocols and applications has provided the worm writers with a potential means of locating new victims in various ways. We present here three strategies that could be used to find the targets in P2P networks.

Passive: Passive P2P worms attach themselves to shared files and wait for potential victims to discover and download them. For example, the Banjamin worm [33] spreads only from and to computers where KaZaa clients are installed. It changes the directory catalog that registers

as the directory accessible to all KaZaa users. Then it copies itself in that directory under many different names, such as popular music, movie, and software titles.

Reactive: Reactive worms, also called contagion worms, propagate parasitically along with normally communication. A reactive P2P worm on a client could probe and infect other peers, when the client establishes new connections. It is report that the reactive P2P worms can infect roughly 9 millions Kazaa clients within one month [3].

Proactive: Proactive worms achieve much faster propagation speed by directly connecting to other peers, whose addresses were cached after previous file requests and downloads. However, these worms may cause more network anomalies than passive or reactive worms.

Crawl: Some open source P2P clients, such as Gnutella and BitTorrent, can be explored to generate hit list. The modified clients can crawl the network actively to discover as many hosts as possible in the P2P network. For example, within 2 hours, 15 crawlers can gather 90,000 unique IP addresses in a Gnutella network [34].

4.2 Modeling P2P Worms

Modeling P2P worms can help us to understand the factors affecting the spread of P2P worms. Most previous studies adopt epidemic model to study P2P worm propagation, emphasizing on propagation process and environment.

Propagation process: To understand how a worm duplicates itself from one host to another, we could use host state transition to describe the propagation process of the worm. Two states, *Susceptible* and *Infected*, are considered in a simple passive P2P worm model [35]. Susceptible is the state that peers are vulnerable for worm; Infected is the state that peers have been infected and start to propagate. Two additional states, *Exposed* and *Recovery*, are added in SEI and SEIR models [36]. Exposed is the state that peers have downloaded one or more infected files, but have not executed them; Recovery stands for peers have installed patch and no longer infect other peers. TF-SEI (Two-Factor-SEI) model considers a *Quarantine* state which stands for the vulnerable peers are immunized by anti-worms and no longer susceptible to infection. Both on-line/off-line behaviors are discussed in [37]. The case that some peers become infected, but never detect the problem and not take any actions to remove the viruses is also considered in that model.

Propagation environment: P2P worm may behave differently when it propagates in different P2P applications. In [38], the authors argue that the assumption [37] that a vulnerable peer can be infected by any of the infected ones is not true in Gnutella like networks since the potential victims for an infected peer are limited to TTL hops away from it and not the entire P2P network. To address the problem, the authors quantify the average number of peers within TTL hops from any given peer and incorporate the neighborhood information into the final model for malware spread. In BT network, Luo, et al. [39] propose an efficient worm in BT network, which finds potential victims by requesting the tracker. They also formulate a model to analyze the effect of worm parameters, such as TTL, on the worm propagation.

4.3 P2P Worm Containment

P2P worm containment methods focus on slowing down the worm propagation speed. There are three methods for P2P worm containment.

Self-defense infrastructure: Self-defense infrastructure is a security P2P architecture that can prevent or isolate P2P worms. Two methods are proposed to design such an architecture in P2P networks.

- *Guardian nodes* can be set to automatically detect P2P worms [40]. These nodes will generate and disseminate alarm messages, called *alerts*, if they identify malicious code inside P2P applications. Upon receiving alerts, an ordinary peer can take several actions to protect itself.
- *Software diversity* can be also used to slow down P2P worm propagation. Since most worms can not run on multiple platforms, we can design a special topology to isolate worms, where peers running on the same platform are not connected. A protocol, called Verme, is designed in [41], which is an extension of Chord. In Verme, all the peers are divided into different groups, called *islands*, according to the type of vulnerabilities. Any two islands with the same vulnerabilities will not be next to each other. In [42], the authors give a model to study the effect of software diversity on P2P worm propagation. Unfortunately, these schemes are not practical, because the software diversity is not very common in P2P networks. It is also hard to believe that the service providers are willing to develop several implementations of the same protocol.

Worm-anti-worm: The anti-worm spreads itself using the same mechanism as the malicious worm [43]. Then, it pathes peer's vulnerability and uses the affected peer to find other vulnerable peers. Although anti-worms have the ability to spread just as fast as regular worms, they have some limitations. One is that the spread of anti-worms may consume a lot of bandwidth. The other is that they run on a system without the system's owner's consent. It is important to note that most computer crime laws do not distinguish "worms" from "anti-worms".

Feedback control: The feed back scheme [44] can not stop P2P worms but slow down them by delaying a peer's requests when it tries to make connections at very high rate. Regrettably, in P2P environment,it is difficult to distinguish abnormal behaviors from normal ones, because the download speed of a peer is not constant. Sometimes, a normal peer may also attempt to connect more peers to increase its QoS.

5. Email Worms

Email worms have had great success propagating them-

selves because they find next victims either by raiding a user's email address book or by searching through the users mailbox [45]. Such addresses are almost certain to be valid, permitting the worm to hijack the user's social web and exploit trust relationships. Some email worms will wait for the user to send a message and attach themselves to it, but, in most cases, they will craft its own message to send to the victim. Some famous email worms include: Melissa in 1999, "Love Letter" in 2000, "W32/Sircam" in 2001, "SoBig" series in 2003, "MyDoom" in 2004, "W32/Blackmal" in 2006 et al.

5.1 Modeling and Analysis of Email Worms

The previous work on modeling and analysis of email worms can be classified into two categories: modeling propagation and statistic analysis.

In the email worm propagation research area, a comprehensive analytical models are given to study email worm behavior based on Interactive Markov Chains. At each node, three statuses are considered: susceptible, infected, and immune. Zou et al. [46] simulate email worm propagation on power law, small world and random graph topologies. Briesemeister et al. [47] study the strategies worms use to infect a system and propagate across a network, and simulate epidemic spreading in scale-free networks.

In a statistic analysis of email worms, Newman et al. [48] develop in detail the theory of random graphs with arbitrary degree distributions, which can be used to derive how many percentage of computers need to be immunized in order to prevent an email worm from spreading out. In the email topology study, Zou et al. [46] show that the size of email groups in Yahoo has a power law distribution. Newman et al. [49] analyze email address book data gathered from a large university computer system and show that the email topology has an exponential and a stretched exponential distribution for in-degree and out-degree, respectively. Wong et al. [50] focus on studying the fundamental behavior and characteristics of two mass-mailing worms, SoBig and MyDoom. By analyzing the trace data, they find that both of them have noticeable abnormalities in the traffic of the infected hosts, and SoBig causes more outgoing infection attempts per infected host than MyDoom does.

5.2 Email Worm Detection

The basic methods of email worm detection are monitoring anomalies caused by such worms. Previous research can be roughly classified into three categories.

Monitoring host traffic: The abnormal behaviors of email worm on a host can be used to generate worm signatures. An ACT (attachment chain tracing) scheme [51] is proposed to detect and control email worms, which is based on the transmission chain concept. This scheme classifies hosts into different categories according to their symptoms, which is defined to be sending emails with attachments at a very high rate, and contact links with other hosts. Martin et al. [52] build models of user behavior by using feature generation on outgoing email traffic. These features can help us to capture the difference between normal and abnormal email activity. Two simple algorithms, TreeCount and SenderCount, are presented in [53], which can be used to detect and generate worm payload signatures.

Mining DNS data: Mining DNS data can detect some massive mailing worms, like MyDoom.A, because those worms rely on DNS server to launch an attack. Ishibashi et al. [54] propose a scheme that uses a Bayesian filter to score query content and hosts. The experimental study shows that it can reduce 89% of mail exchange queries. Musashi et al. [55] present a similar scheme to indirectly detect email worms by investigating the log files in DNS. They find that the DNS records of infected hosts are different from that of normal hosts.

Using honeypot and honeynet: Honeypot and honeynet can be utilized to detect "zero day" email worms in enterprise networks. Zou et al. [56] present a feedback email worm defense system in enterprize network, which uses honeypot to early detect the presence of an email worm in the Internet. Another dynamic architecture [57] is also proposed to defend against email worms. This architecture integrates several concepts, including Markov decision processes, Rabin fingerprinting and honeypots, to quarantine unknown email worms.

6. IM Worms

By exploiting features and vulnerabilities of IM clients and protocols, IM worms could generate automated file transfer requests or send malicious URLs using a compromised user's IM user ID. Examples of notorious IM worms include "Choke" in 2001, "SoFunny" in 2001, "JS Menger" in 2005, "Bropia/Kelvir" in 2005, and "Serflog" in 2005 et al.

There are only a few methods for IM worm detection and containment. A straightforward solution [58] is to shut down the IM server temporarily, manually analyze the worm and build a client patch, then force users to update when they login as the IM server comes alive again. Smith et al. [59] propose a scheme that disconnect the most-connected users to slow down the spread and allow time to build and apply a patch. Williamson et al. [60] introduce a mechanism to restrict IM worms by capturing the deference between worm-spreading network traffic and normal traffic on many Internet protocols. To avoid the shortcomings of the above solutions, Mannan et al. [61] defend against IM worms by using CAPTCHA to restrict automatically generated file transfer requests and text messages with malicious URLs. Yan et al. [62] apply change-point detection techniques to detect both fast scanning and self-restraining IM worms. Zhang et al. [63] implement a P2P-IM-worm which can build a simple http server on the infected host, and propagate itself using P2P patten. They also give us some tips for avoiding infection.

7. Research Trends

After a survey of worms, in terms of concepts, classification, characteristics and the state-of-the-art defense mechanism, we think there are the following trends for worm research.

- The lines between various worm types are becoming blurred, a worm may use many propagation methods. Correspondingly, the defense side should use a hybrid defense method, such as a combination of signature-based and anomaly-based schemes, to cope with these "hybrid" worms.
- Worms constantly evolve, becoming faster and stealthier. For example, permutation scanning worms are very likely to appear in the future, though they remain a potential threat (there are no real permutation-scanning worms has been reported in the Internet yet). Also for the polymorphic worms and metamorphic worms. The defense of these fast and stealthy worms is an challenging problem.
- As we have introduced in previous sections, some new defense techniques, such as automatic signature generation, honeypot, network-level simulation, are promising and likely to be widely used in the future for worm detection.

8. Conclusion

In this article, we introduce the concept of various worms and classify them into four types based on which space a worm finds a target. Since most existing worms are Internet worms, we pay most attention on them– we outline the characteristics of Internet worms with respect to their target finding strategies, propagation methods, and anti-detection techniques. We also introduce the state-of-the-art worm detection and containment methods against Internet worms. For P2P, Email and IM worms, we briefly present their characteristics, defense methods and related research work.

Since more and more worms act as carriers for other malwares, worm defense remains to be a critical issue for network security in the future. Besides the traditional directions for worm research, like exposure and study of ad-hoc worms, modeling and analysis of worm propagation, worm detection and containment, we need to prepare for the coming threats, such as flash worms, polymorphic worms and P2P worms. So far, since there are no perfect solutions for worm defense, defense in-depth (defense in all levels) could be our best choice.

References

[1] D.M. Kienzle and M.C. Elder, "Recent worms: A survey and trends," WORM '03: Proc. 2003 ACM Workshop on Rapid Malcode, pp.1–10, 2003.
[2] J. Evers, "Instant messaging worm exploits jpeg vulnerability," 2004. http://www.computerworld.com/securitytopics/security/holes/story/0,10801,96268,00.html
[3] S. Staniford, V. Paxson, and N. Weaver, "How to own the Internet in your spare time," Proc. 11th USENIX Security Symposium, pp.149–167, USENIX Association, 2002.
[4] C.C. Zou, D. Towsley, W. Gong, and S. Cai, "Routing worm: A fast, selective attack worm based on ip address information," Proc. PADS '05, pp.199–206, Washington D.C., 2005.
[5] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, "A taxonomy of computer worms," WORM '03: Proc. 2003 ACM Workshop on Rapid Malcode, pp.11–18, ACM Press, 2003.
[6] N. Provos, J. McClain, and K. Wang, "Search worms," WORM '06: Proc. 4th ACM Workshop on Recurring Malcode, pp.1–8, 2006.
[7] Z. Zhu, G. Lu, Y. Chen, Z.J. Fu, P. Roberts, and K. Han, "Botnet research survey," Proc. COMPSAC '08, pp.967–972, Washington D.C., 2008.
[8] M. Polychronakis, K.G. Anagnostakis, and E.P. Markatos, "Emulation-based detection of non-self-contained polymorphic shellcode," RAID, pp.87–106, 2007.
[9] Q. Zhang, D.S. Reeves, P. Ning, and S.P. Iyer, "Analyzing network traffic to detect self-decrypting exploit code," Proc. 2nd ACM Symposium on Information, Computer and Communications Security, pp.4–12, 2007.
[10] D. Brumley, J. Newsome, D. Song, H. Wang, and S. Jha, "Towards automatic generation of vulnerability-based signatures," 2006 IEEE Symposium on Security and Privacy, pp.2–16, 2006.
[11] P. Li, M. Salour, and X. Su, "A survey of internet worm detection and containment," IEEE Communications Surveys and Tutorials, vol.10, no.1, pp.20–35, 2008.
[12] S. Singh, C. Estan, G. Varghese, and S. Savage, "Automated worm fingerprinting," 6th USENIX OSDI, 2004.
[13] T. Cymru, "The darknet project," 2008. http://www.team-cymru.org/Services/darknets.html
[14] V. Yegneswaran, P. Barford, and D. Plonka, "On the design and use of internet sinks for network abuse monitoring," Proc. RAID, pp.146–165, 2004.
[15] D. Dagon, X. Qin, G. Gu, W. Lee, J. Grizzard, J. Levine, and H. Owen, "Honeystat: Local worm detection using honeypots," RAID, pp.39–58, 2004.
[16] P. Niels, "Honeyd - A virtual honeypot daemon," 2003. http://www.honeyd.org
[17] Y. Tang, H.P. Hu, X.C. Lu, and J. Wang, "Honids: Enhancing honeypot system with intrusion detection models," Fourth IEEE International Workshop on Information Assurance (IWIA'06), pp.135–143, 2006.
[18] C. Leita, K. Mermoud, and M. Dacier, "Scriptgen: An automated script generation tool for honeyd," ACSAC '05: Proc. 21st Annual Computer Security Applications Conference, pp.203–214, 2005.
[19] M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A.C. Snoeren, G.M. Voelker, and S. Savage, "Scalability, fidelity, and containment in the potemkin virtual honeyfarm," SIGOPS Oper. Syst. Rev., vol.39, no.5, pp.148–162, 2005.
[20] J. Newsome, B. Karp, and D. Song, "Polygraph: Automatically generating signatures for polymorphic worms," 2005 IEEE Symposium on Security and Privacy, pp.226–241, 2005.
[21] M.V. Mahoney, "Network traffic anomaly detection based on packet bytes," SAC '03: Proc. 2003 ACM Symposium on Applied Computing, pp.346–350, 2003.
[22] K. Wang, J.J. Parekh, and S.J. Stolfo, "Anagram: A content anomaly detector resistant to mimicry attack," RAID, pp.226–248, 2006.
[23] K. Wang, G. Cretu, and S.J. Stolfo, "Anomalous payload-based worm detection and signature generation," Recent Advances in Intrusion Detection (RAID), 2003.
[24] C. Krügel, T. Toth, and E. Kirda, "Service specific anomaly detection for network intrusion detection," SAC '02: Proc. 2002 ACM Symposium on Applied Computing, pp.201–208, 2002.
[25] J. Newsome and D. Song, "Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software," 12th Annual Network and Distributed System Security

Symposium, 2005.
[26] J. Xu, P. Ning, C. Kil, Y. Zhai, and C. Bookholt, "Automatic diagnosis and response to memory corruption vulnerabilities," 12th ACM Conference on Computer and Communications Security, pp.223–234, 2005.
[27] Z. Liang and R. Sekar, "Fast and automated generation of attack signatures: A basis for building self-protecting servers," 12th ACM Conference on Computer and Communications Security, pp.213–222, 2005.
[28] X.F. Wang, Z. Li, J. Xu, M.K. Reiter, C. Kil, and J.Y. Choi, "Packet vaccine: Black-box exploit detection and signature generation," 13th ACM Conference on Computer and Communications Security, pp.37–46, 2006.
[29] Z. Li, M. Sanghi, Y. Chen, M.Y. Kao, and B. Chavez, "Hamsa: Fast signature generation for zero-day polymorphic worms with provable attack resilience," 2006 IEEE Symposium on Security and Privacy, 2006.
[30] Y. Tang, X. Lu, and B. Xiao, "Generating simplified regular expression signatures for polymorphic worms," 4th International Conference on Autonomic and Trusted Computing (ATC-07), 2007.
[31] S. Chen and Y. Tang, "Daw: A distributed antiworm system," IEEE Trans. Parallel Distrib. Syst., vol.18, no.7, pp.893–906, 2007.
[32] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, and D. Watson, "The Internet motion sensor: A distributed blackhole monitoring system," Proc. 12th ISOC Symposium on Network and Distributed Systems Security (NDSS), pp.167–179, 2005.
[33] M. Singer, "Benjamin worm plagues kazaa," Tech. Rep., siliconvalley.internet.com, 2002.
[34] J.K.K. Lakshminarayanan, "Implications of peer-to-peer networks on worm attacks and defenses," Tech. Rep., UCB, 2003.
[35] J. Ma, X. Chen, and G. Xiang, "Modeling passive worm propagation in peer-to-peer system," International Conference on Computational Intelligence and Security, pp.1129–1132, 2006.
[36] Y. Yao, X. Luo, F. Gao, and S. Ai, "Research of a potential worm propagation model based on pure p2p principle," ICCT '06: International Conference on Communication Technology, pp.1–4, 2006.
[37] R. Thommes and M. Coates, "Epidemiological modeling of peer-to-peer viruses and pollution," INFOCOM'06, 2006.
[38] K. Ramachandran and B. Sikdar, "Modeling malware propagation in gnutella type peer-to-peer networks," IPDPS'06: Parallel and Distributed Processing Symposium, 2006.
[39] J. Luo, B. Xiao, G. Liu, Q. Xiao, and S. Zhou, "Modeling and analysis of self-stopping bt worms using dynamic hit list in p2p networks," SSN'09, 2009.
[40] L. Zhou, L. Zhang, F. Mcsherry, N. Immorlica, M. Costa, and S. Chien, "A first look at peer-to-peer worms: Threats and defenses," Proc. IPTPS'05, 2005.
[41] F. Freitas, R. Rodrigues, C. Ribeiro, P. Ferreira, and L. Rodrigues, "Verme: Worm containment in peer-to-peer overlays," IPTPS'07: Proc. 6th International Workshop on Peer-to-Peer Systems, 2007.
[42] Y. Zhou, Z. Wu, H. Wang, J. Zhong, Y. Feng, and Z. Zhu, "Breaking monocultures in p2p networks for worm prevention," Proc. Fifth International Conference on Machine Learning and Cybernetics, 2006.
[43] Y. Yao, L. Wu, F. Gao, W. Yang, and G. Yu, "A waw model of p2p-based anti-worm," ICNSC'08: IEEE International Conference on Networking, Sensing and Control, pp.1131–1136, 2008.
[44] K. Wu and Y. Feng, "Proactive worm prevention based on p2p networks," IJCSNS'06: International Journal of Computer Science and Network Security, 2006.
[45] E. Levy, "Worm propagation and generic attacks," IEEE Security and Privacy, vol.3, no.2, pp.63–65, 2005.
[46] C. Zou, D. Towsley, and W. Gong, "Email virus propagation modeling and analysis," Tech. Rep., Umass ECE Dept., 2003.
[47] L. Briesemeister, P. Lincoln, and P. Porras, "Epidemic profiles and defense of scale-free networks," WORM'03: Proc. ACM CCS Workshop on Rapid Malcode, 2003.
[48] M.E.J. Newman, S.H. Strogatz, and D.J. Watts, "Random graphs with arbitrary degree distribution and their applications," Tech. Rep., Santa Fe Institute, 2000.
[49] M. Newman, S. Forrest, and J. Balthrop, "Email networks and the spread of computer viruses," Phys. Rev. E, vol.66, pp.200–202, 2002.
[50] C. Wong, S. Bielski, J.M. McCune, and C. Wang, "A study of mass-mailing worms," WORM' 04: Proc. ACM Workshop on Rapid Malcode, pp.1–10, 2004.
[51] J. Xiong, "Act: Attachment chain tracing scheme for email virus detection and control," WORM' 04: Proc. ACM Workshop on Rapid Malcode, 2004.
[52] S. Martin, A. Sewani, B. Nelson, K. Chen, and A.D. Joseph, "Analyzing behaviorial features for email classification," Proc. CEAS '05, Stanford University, July 2005.
[53] P. Gopalan, K. Jamieson, P. Mavrommatis, and M. Poletto, "Signature metrics for accurate and automated worm detection," WORM'06: Proc. ACM Workshop on Rapid Malcode, 2006.
[54] K. Ishibashi, T. Toyono, and K. Toyama, "Detecting mass-mailing worm infected hosts by mining dns traffic data," SIGCOMM05 Workshops, 2005.
[55] Y. Musashi, R. Matsuba, and K. Sugitani, "Indirect detection of mass mailing worm-infected pc terminals for learners," ICETA'04: International Conference on Emerging Telecommunications Technologies and Applications, 2004.
[56] C.C. Zou, W. Gong, and D. Towsley, "Feedback email worm defense system for enterprise networks," Tech. Rep., Umass ECE Dept., 2004.
[57] T.M. Mahmoud, A.S. Ehab, and B. Raouf, "An architecture for an email worm prevention system," Securecomm and Workshops, pp.1–9, 2006.
[58] N. Hindocha, "Threats to instant messaging," Tech. Rep., Symantec Security Response, 2003.
[59] R.D. Smith, "Instant messaging as a scale-free network," 2002, http://www.citebase.org/
[60] M. Williamson, "Throttling viruses: Restricting propagation to defeat malicious mobile code," ACSAC'02: Proc. 18th Annual Computer Security Applications Conference, pp.61–68, 2002.
[61] M. Mannan and P.C. van Oorschot, "On instant messaging worms, analysis and countermeasures," WORM'05: Proc. 2005 ACM Workshop on Rapid Malcode, pp.2–11, 2005.
[62] G. Yan, Z. Xiao, and S. Eidenbenz, "Catching instant messaging worms with change-point detection techniques," Proc. 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats, pp.1–10, 2008.
[63] G. Zhang and F.M. Kugblenu, "The feasibility of p2p technique used in im worm," Tech. Rep., Computer Science Dept., Blekinge Institute of Tech., 2006.

Yong Tang received the B.Sc and Ph.D. degrees in computer science from College of Computer, National University of Defense Technology, China, in 2002 and 2008 respectively. Now he is a lecturer in the College of Computer, National University of Defense Technology, China. His primary research field is network security, especially on Internet worm attacks, intrusion detection and Honeypot.

Jiaqing Luo received B.Sc and M.Sc degrees in computer science from School of Computer Science and Engineering, University of Electronic Science and Technology of China, in 2000, 2004 respectively. Now he is a Ph.D student in the Department of Computing, The Hong Kong Polytechnic University. His research interests include P2P security and P2P search.

Bin Xiao received the B.Sc and M.Sc degrees in Electronics Engineering from Fudan University, China, in 1997 and 2000 respectively, and Ph.D. degree from University of Texas at Dallas, USA, in 2003 from Computer Science. Now he is an Assistant Professor in the Department of Computing of Hong Kong Polytechnic University, Hong Kong. His research interests include communication and security in computer networks, peer-to-peer networks, wireless mobile ad hoc and sensor networks.

Guiyi Wei received the B.Sc and M.Sc degrees in Information Management and Information System from Zhejiang Gongshang University, China, in 1996 and 2000 respectively, and Ph.D. degree from Zhejiang University, China, in 2006 from Computer Science. Now he is an Associate Professor in the Department of Computer and Information Engineering of Zhejiang Gongshang University, China. His research interests include grid computing, peer-to-peer networks, wireless sensor networks.