Worm Poisoning Technology and Application

Bing Wu[†], *Xiaochun Yun*[†] *and Xiang Cui*[††],

[†]Harbin Institute of Technology, Harbin, P.R.China, [††]CNCERT/CC, BeiJing, P.R.China

Summary
In this paper, the concept of Worm Poisoning and PoisonWorm are presented and the feasibility of Worm Poisoning is testified. A propagation model named SIRP Model and PoisonWorm's side-effect on network traffic are given and compared with the classical epidemic Kermack-Mckendrick model. The feasibility and necessity of PoisonWorm and its application are highlighted in active defense system against Internet worms. In addition, the technology of P2P-based unknown worm detection and signature verification are briefly introduced.

Key words:
Poisoning, PoisonWorm, SIPR, Detection, DHT, P2P

Introduction

Current strategy against Internet worms is similar to capturing mouse with mousetrap, that is, to clip the randomly passing mouse and never release it until it dies. However, this strategy is less effective than that of spreading pest control chemicals to start a plague among cockroach group. For the infected cockroach, it is not expected to die at once but to go back to its nest and infect others, killing them at an exponential rate. The theory of Worm Poisoning is similar to the pest-toxicant production techniques. The PoisonWorm functions like the pest-toxicant and the poisoned worm is comparable to the infected pest.

In the research field of automatic unknown scanning worm detection and signature creation, there are many successful outcomes, including AutoGraph[12], EarlyBird[8], HoneyComb[11], NetBait[13], DSC[24] etc. DSC(Destination-Source Correlation) discovers unknown worm based on local host activity, other research is based on mass-scanning, unused ip accessing, failed connection, DNS query[14] etc. AutoGraph, EarlyBird, HoneyComb, NetBait can extract signature based on large amount packets. PoisonWorm makes use of these outcomes unchanged.

In the research field of worm containment, there are also many outcomes, including modifying local TCP/IP protocol stack to limit outgoing connection speed[1], using worm-hole and Honeynet to slowing down worm spread speed, filtering blacklist and content by Firewall plus Router[2,3], and Anti-wormetc. Most of these control mechanism must deploy hard and soft equipments widely which is costly. Due to the fact that Internet is designed to have very strong connectedness, it is hard to cut down all spread paths. There is P2P-based IDS [23] abroad, but it is very different from the technology in the paper.

Worm Poisoning is a newly-invented technology. It tricks the malicious worms to spread irrelevant file or code through their own mechanisms. The worm which poisons others and propagates infection is called PoisonWorm. So PoisonWorm is a special worm with active spread motivation, but have not self-propagating capability, it can obtain spread ability when some other malicious worms break out, and will reduce the negative influence of the malicious worm gradually, without causing extra burden to the Internet or its host. Usually, PoisonWorm is latent in the host. When it detects malicious worms, it will try to trick the worm to spread PoisonWorm file. If there're N worms in the host, the spread speed and the number of infected victims of PoisonWorm are the union of the N worms.

1. The Feasibility of Worm Poisoning Technology

There are various kinds of spread mechanism of worms. Whether the common characteristic of worms can be extracted is the main concern. Firstly, PoisonWorm would not carry a signature library like normal AV software, and also does not limit to poisoning the fast scanning worm.

1.1 Worm Propagation Procedure

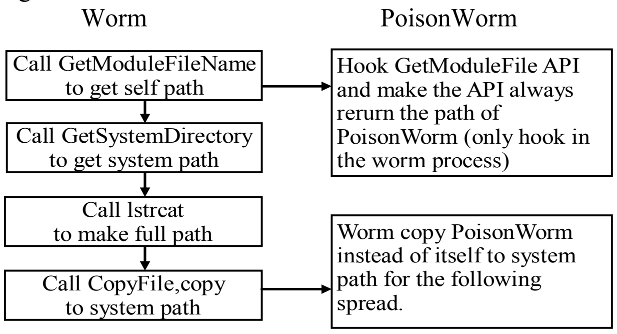
Worm may spread using multi-vectors such as vulnerability, backdoor, e-mail, cracking simple password, IM and P2P etc, of which there are no common characteristics in the spread mechanism. However, the residence mechanism of worm in the victim host has something in common. The commonness, with which this paper concerned, is different from the usual anti-virus (which is based on the virus behavior, like API function, executive order and so on) detection technology.
Worm always resides in the compromised victim in order to have itself executed when the OS restarts. It generally follows the steps below:
a. Get its path by calling the 'GetModuleFileName' API; b. Get system or windows directory path by calling the 'GetSymtemDirectory' or 'GetWindows Directory' API; c. Link system path and worm name to establish the full path by calling 'lstrcat' API; d. Copy itself to the full path by

Manuscript received June 5, 2006.
Manuscript revised June 25, 2006.

calling 'CopyFile' API; e. Modify the Registry or other AutoStart file, which may enable the autorun function; f. Begin to propagate. This is the general procedure after the worm is executed.

1.2 PoisonWorm Propagation Procedures

PoisonWorm makes use of the first step during which the worm calls GetModule FileName to get self-path. The purpose of so doing is to return the path of the running worm process. PoisonWorm hooks the API so that it always returns to PoisonWorm's file path by modifying the address space of the poisoned worm's process. So there is no influence on other normal processes. After the malicious worm calls the API, it will get PoisonWorm's path, copy PoisonWorm to the destination path and makes PoisonWorm autorun. Most importantly worm will spread PoisonWorm instead of itself from now on. The following figure will show what PoisonWorm will do:


Worm
PoisonWorm
Call GetModuleFileName to get self path
Hook GetModuleFile API and make the API always rerurn the path of PoisonWorm (only hook in the worm process)
Call GetSystemDirectory to get system path
Call lstrcat to make full path
Worm copy PoisonWorm instead of itself to system path for the following spread.
Call CopyFile,copy to system path

PoisonWorm is effective on the following type of worms:

Exploiting Worm: The MSBlaster worm is a typical buffer overflow worm. Procedures are shown below: a. get self path by calling GetModuleFileName; b. after exploiting remote system successfully, the shellcode binds a cmd shell in the remote system, open tcp 4444 port and listens. c. send tftp –i localip GET msblast.exe string as a cmd to remote tcp 4444 port. d. remote system runs the tftp command and connects back to the tftp server running in the infection source; e. the tftp server will transfer the MSBlaster file. Because PoisonWorm has been hooked to GetModuleFile Name API, MSBlaster will transfer PoisonWorm to remote instead itself. Now we have proved that MSBlaster spreads PoisinWorm with its own spread mechanism. Similarly, this mechanism of the wrok works also on Nimda, Welchia, Sasser etc.

E-Mail worm: For the example of Mydoom and Beagle. a. get self path by calling GetModuleFileName. Copy to system path and save the returned path by GetModuleFileName to a global string variable e.g. G; b. open the file G and encode the file for the purpose of spread as an email attachment; c. search email addresses, query for the MX record of DNS to get the SMTP Server. Send the email with the worm attachment; Because PoisonWorm has been hooked to GetModuleFileName API, Mydoom and Beagle will encode PoisonWorm and send it to victims via email attachment.

P2P-based worm: This type of worms calles GetModuleFileName to get self path, then copy itself to the shared directory of P2P software such as Kazaa. Actually, they are easier than other worms.

Memory-Residence Worm: This type of worms does not need to call GetModuleFileName API because they have no corresponding file on the hard disk. They just exist in the RAM. In this case, PoisonWorm won't work. But we can make use of PoisonWorm's extended attribute to contain them.

Packed worm: Many worms with file carrier are packed with packing software such as UPX. Thus they can not only avoid infection of normal virus but also reduce the body size. PoisonWorm is also effective on packed worm because it does not infect worms in File, but in memory. GetModuleFileName is an export function of Kernel32.dll. Kernel32.dll is loaded with OS at the early boot stage. Other processes use the API by sharing the loaded Kernel32.dll. The address of the API will not change before or after the packed worms remove their package. PoisonWorm modifies the GetModuleFileName API in the memory space of the packed worm. So the file change of packed worm does no matter.

Other worm: Worms that use rootkit or crack password are similar to Exploiting Worm. IM-based worms are similar to E-Mail Worm.

So far we have proved PoisonWorm can poison all kinds of worms except Memory-Residence worm. The poisoned worms use their own spread mechanism by spreading the file of PoisonWorm and not poisoned worm itself.

Of course, we can use other technologies to do the job, for example, reusing the port, which is used to transport file such as UDP 69 port of MSBlaster and TCP 5554 port of sasser worm. Thus, when these worms' remote shellcode connects with download file, PoisonWorm accepts the connection and transports itself to the remote victim.

1.3 Initialization of PoisonWorm

PoisonWorm can firstly spread using worms captured by honeypots. An unpatched windows 2000 system will be infected by malware in 25 minutes on average [16] after connected to the Internet. By several honeypots, PoisonWorm can reach many hosts which have been infected by active worms and bots.

After a short period, PoisonWorm can spread in a very large range of Internet. These PoisonWorms have the ability of finding new worms and receiving control commands.

1.4 How PoisonWorm detects the worms to poison?

PoisonWorm is not AV software so it can only deal with wide-spread worms. What PoisonWorm tries to save is not the single infected system but the entire Internet. For the known wide-spread worms, PoisonWorm recognizes them by filename and path, file or signature. These signatures and policies are distributed over PoisonWorm via plugins, PoisonWorm Command and Control Center which consists of security expertise and servers. For the unknown worms, PoisonWorm can use existing technology such as AutoGraph[12], EarlyBird[8], HoneyComb[11], NetBait[13], DSC[24] to track them. It can also use "DHT-Based UNKNOWN-WORM Detection and Signature Verification" to accelerate the speed of detection.

To summarize, Worm Poisoning is effective for majority worms such as CodeRed, Slammer. Witty has no file carrier, but can be controlled and be quarantined after PoisonWorm finds them.

2. The extend attribute of PoisonWorm

In order to serve "Worm Active Defense System", PoisonWorm must add some extended attribute listed below:

2.1 Controllability

The meaning and aim of PoisonWorm is not to kill particular worm, but to exist for a long time as a part of Worm Active Defense System. So it must be controlled safely. "MD5+TimeStamp"[17] is an effective way to solve the problem. Every PoisonWorm will carry the same hard-coded MD5 hash. Upon receiving a command such as updating signature, it calculates the hash of the command header and compare with its own. If they match, it will accept and execute the following command. The command will instruct PoisonWorm what to execute, what functions to add or what new signature to update etc. To avoid the command reused by attacker and by sniffer, PoisonWorm checks the TimeStamp each time. Thus one command can only be executed for once. Furthermore, a new MD5 hash is embedded in the command to be used next time. PoisonWorm must replace the old MD5 with the newly obtained one every time.

We can use other ways such as public/private key to control: every PoisonWorm carries a public key. PoisonWorm controller has the corresponding private key. PoisonWorm only accepts command signed by the private key. The shortcoming is that it needs too much code and work.PoisonWorm must support plugin, so it can add new function and remove unnecessary one just like some well-designed bots.

2.2 Detecting unknown worm speedy and precisely by DHT-Based technology

PoisonWorm should not have central control point. The PoisonWorm uses P2P network architecture and DHT (Distribute Hash Table) to query information. To improve the speed and precision of unknown worm detection, one PoisonWorm must communicate with the rest worms in the system.

In the article "DHT-Based UNKNOWN-WORM Detection and Signature Verification", a method is adopted to improve the coordination policy of Earlybird and Autograph:

a. PoisonWorm uses EarlyBird or DSC to find suspicious flow, and Autograph to create signature. PoisonWorm also find suspicious file by heuristic technology. Since all the unknown worm detection technology is not 100% reliable, PoisonWorm must verify the result. PoisonWorm calculates the hash of the signature or the file as the Object to query.

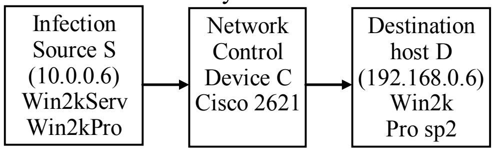
b. Some P2P software use DHT to find particular files. We apply this idea to worm detection. In order to verify whether the signature and suspicious file appear in a single host or widely in the Internet, PoisonWorm sends queries to its peer PoisonWorm. The queried object is the hash of signature or suspicious file. Most of peer PoisonWorms have not the object, perhaps it is a scan or spam behavior of local host. Strictly, every PoisonWorm should sends more than one query out because spread of worm need time interval. If the query results increase just like the same characteristic of worm propagation, the probability of a wide-spread worm outbreak is very high. This idea eliminates the influence of background traffic noise greatly.

2.3 Robustness

PoisonWorm should have the ability to defend simple improvements of worms by VXers, for example, to call lower-level functions than GetModuleFileName to get self path.

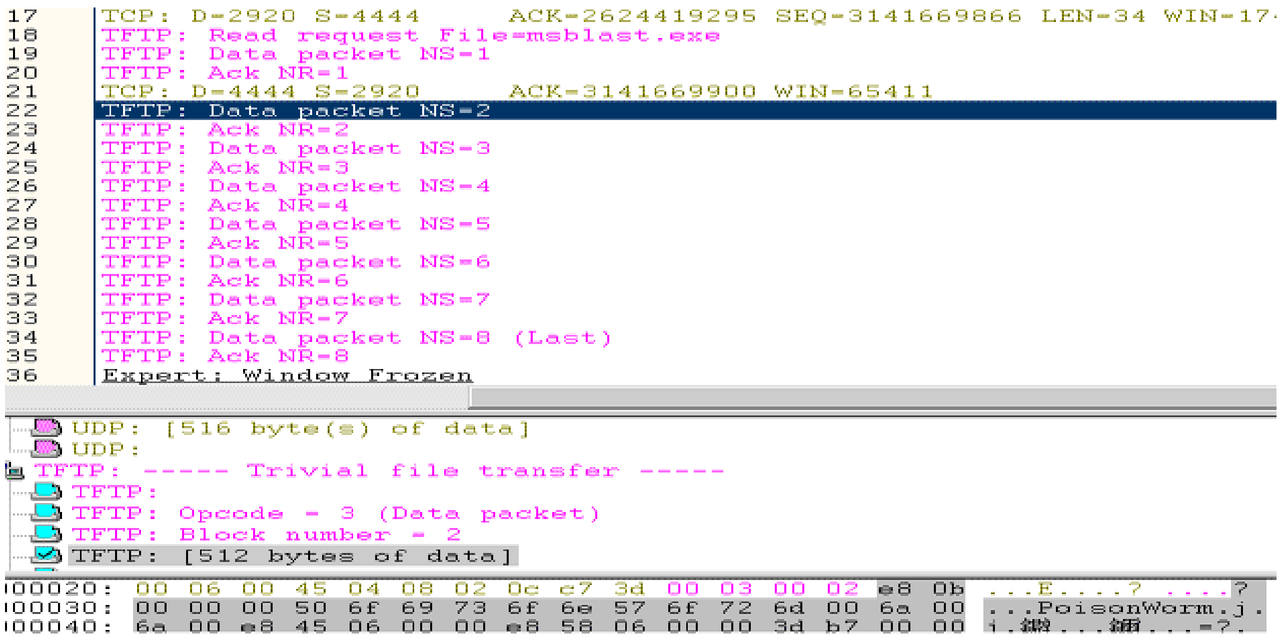
3. Test Result

PoisonWorm.asm was compiled in masm32 v8 under Win2kPro OS. The source code has 480 lines and the compiled exe file is 3584 bytes.


Infection Source S (10.0.0.6) Win2kServ Win2kPro
Network Control Device C Cisco 2621
Destination host D (192.168.0.6) Win2k Pro sp2

3.1 Test environment

First, running MSBlaster, Sasser and PoisonWorm in turn in S (infection source). D(short for Destination) is a vulnerable host running Win2kPro sp2 OS, C is a Cisco 2621 router which enables D to be infected by worms in S quickly and traffic control. Because the scan policy of worms is different, D need some time to be scanned successfully. To solve the problem of time delay, we configured a DNAT in C. The function of the DNAT is that no matter what the scanned destination IP is, C changes the destination IP to D's IP and send the scan packet to D. Then, the Source IP of the response packet from D is changed to the original scanned IP by S. All the work is done by C obviously.

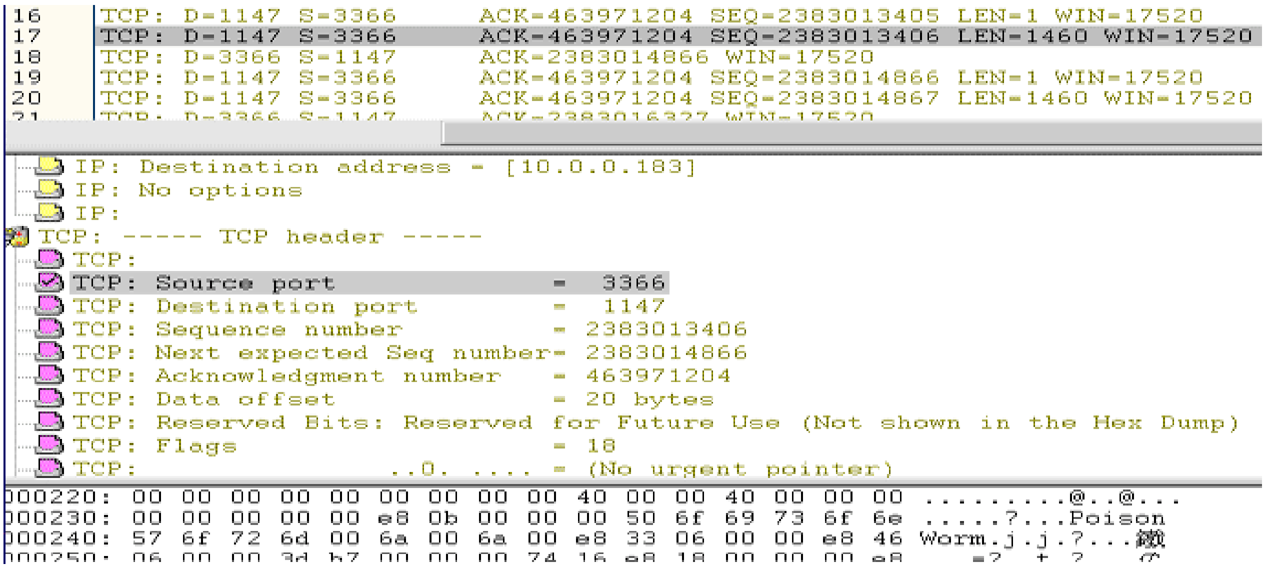


```
17  TCP: D=2920 S=4444      ACK=2624419295 SEQ=3141669866 LEN=34 WIN=17
18  TFTP: Read request File=msblast.exe
19  TFTP: Data packet NS=1
20  TFTP: Ack NR=1
21  TCP: D=4444 S=2920      ACK=3141669900 WIN=65411
22  TFTP: Data packet NS=2
23  TFTP: Ack NR=2
24  TFTP: Data packet NS=3
25  TFTP: Ack NR=3
26  TFTP: Data packet NS=4
27  TFTP: Ack NR=4
28  TFTP: Data packet NS=5
29  TFTP: Ack NR=5
30  TFTP: Data packet NS=6
31  TFTP: Ack NR=6
32  TFTP: Data packet NS=7
33  TFTP: Ack NR=7
34  TFTP: Data packet NS=8 (Last)
35  TFTP: Ack NR=8
36  Expert: Window Frozen

    UDP: [516 byte(s) of data]
    UDP:
TFTP: ----- Trivial file transfer -----
    TFTP:
    TFTP: Opcode = 3 (Data packet)
    TFTP: Block number = 2
    TFTP: [512 bytes of data]

000020: 00 06 00 45 04 08 02 0c c7 3d 00 03 00 02 e8 0b  ...E....? ....?
000030: 00 00 00 50 6f 69 73 6f 6e 57 6f 72 6d 00 6a 00  ...PoisonWorm.j.
000040: 6a 00 e8 45 06 00 00 e8 58 06 00 00 3d b7 00 00  j.鎓...鍎...=?.
```

Fig. 1. Packets sent by MSBlaster worm



```
16  TCP: D=1147 S=3366      ACK=463971204 SEQ=2383013405 LEN=1 WIN=17520
17  TCP: D=1147 S=3366      ACK=463971204 SEQ=2383013406 LEN=1460 WIN=17520
18  TCP: D=3366 S=1147      ACK=2383014866 WIN=17520
19  TCP: D=1147 S=3366      ACK=463971204 SEQ=2383014866 LEN=1 WIN=17520
20  TCP: D=1147 S=3366      ACK=463971204 SEQ=2383014867 LEN=1460 WIN=17520

    IP: Destination address = [10.0.0.183]
    IP: No options
    IP:
TCP: ----- TCP header -----
    TCP:
    TCP: Source port                =  3366
    TCP: Destination port           =  1147
    TCP: Sequence number            = 2383013406
    TCP: Next expected Seq number= 2383014866
    TCP: Acknowledgment number      = 463971204
    TCP: Data offset                = 20 bytes
    TCP: Reserved Bits: Reserved for Future Use (Not shown in the Hex Dump)
    TCP: Flags                      = 18
    TCP:                 ..0. .... = (No urgent pointer)
000220: 00 00 00 00 00 00 00 00 00 40 00 00 40 00 00 00  .........@..@...
000230: 00 00 00 00 00 e8 0b 00 00 00 50 6f 69 73 6f 6e  .....?...Poison
000240: 57 6f 72 6d 00 6a 00 6a 00 e8 33 06 00 00 e8 46  Worm.j.j.?...鐵
000250: 06 00 00 3d b7 00 00 00 74 16 e8 18 00 00 00 e8
```

Fig. 2. Packets sent by Sasser worm

To prevent D from receiving too many packets, we configured TCP source port ACL in C. The ACL only allowed source ports among 2500, 2900, 3400, 3800, 5554(sasser uses it) to pass through. Several seconds after the worms started, the source port could increase to 2500. So PoisonWorm has enough time to poison those worms. When D responded to S, the source port was random but the destination was fixed. So we allowed any destination ports among TCP 4444, UDP 69 (MSBlaster)、TCP 9996, 5554(sasser) to pass through. All these measures enabled D to be infected quickly and reliably.

3.2 Result

Executing MSBlaster and PoisonWorm in turn on host S, D was infected successfully in 10 seconds. PoisonWorm was transferred to host D by MSBlaster and ran steadily. At the same time, MSBlaster and PoisonWorm kept running in host S. It's MSBlaster who transferred PoisonWorm to host D and let it run. The details of transferred data are shown in Fig.1. For The sasser worm tested, we got a similar successful result like MSBlaster shown in Fig.2. The two figures show what the two worms actually transferred in the network: not themselves, but PoisonWorm.

For Welchia (MSBlaster-remover) and NetSky worm, we only tested on local host. Executing PoisonWorm after running Welchia and NetSky, we observed that PoisonWorm was copied to the destination folder by the two worms separately. We can conclude that if the two worms spread based on either the destination path or running process path, they will always spread PoisonWorm.

4. SIPR Model Analysis

4.1 SIPR Model introduction

Table 1. Notations in this SIPR model.

Notation	*Definition*
I(t)	Number of infectious hosts at time t
S(t)	Number of susceptible hosts at time t
R(t)	Number of removed hosts from infectious population at time t
Ps(t)	Number of susceptible hosts infected by PoisonWorm at time t
Pi(t)	Number of Ps hosts infected by malicious worm at time t
β	Pairwise rate of infection in worm propagation model
γ	Removal rate of infectious hosts
k	Self-killing rate of PoisonWorm in Pi
η	Average scan rate per infected host(use 4000)
N	Total number of hosts under consideration (use 1000001)

The SIPR model is presented here in order to analyze Worm Poisoning Technology. As an epidemic model it assumes that each host exists in one of the four possible states: susceptible, infected, poisoned or recovered. It assumes that susceptible hosts can be infected by either the worm or PoisonWorm, and an infected host will develop immunity to both the malicious worm and PoisonWorm. The used notations are listed in table 1 and the meanings of the notations are explained in fig.3.

① PoisonWorm kills the local malicious worm with k probability, so the host converts to Ps (PoisonWorm in susceptible host)
② Ps convert to Pi (malicious worm coexisting with PoisonWorm in susceptible host) when infected by malicious worm
③ susceptible host convert to Ps when infected by PoisonWorm
④⑤ the activity of infection, the malicious worm is transferred
⑥ the activity of infection, the PoisonWorm is transferred
⑦ susceptible host converts to infected host when infected by malicious worm

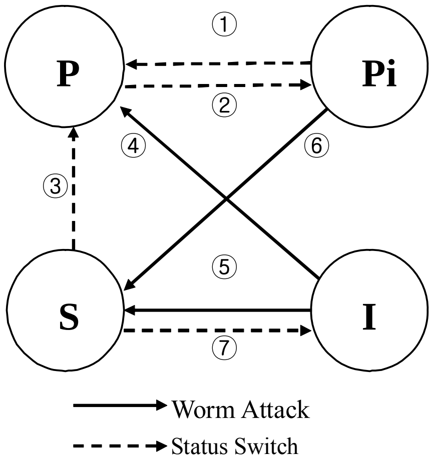


Fig. 3. SIPR model attacking and status switching graph

Based on the above analysis, we can derive the following SIPR model:

$$\begin{cases} dI(t)/dt = \beta I(t)S(t) - dR(t)/dt & (1) \\ dPi(t)/dt = \beta I(t)Ps(t) - k\,Pi(t) & (2) \\ dPs(t)/dt = \beta Pi(t)S(t) - dPi(t)/dt & (3) \\ dS(t)/dt = -\beta[Pi(t) + I(t)]S(t) & (4) \\ dR(t)/dt = \gamma I(t) & (5) \end{cases}$$

Equations (1-5) are five coupled non-linear differential equations, referred to as the SIPR Model.
In comparison, Kermack-Mckendrick(SIR) model is also shown here:

$$\begin{cases} dI(t)/dt = \beta I(t)S(t) - dR(t)/dt \\ dS(t)/dt = -\beta I(t)S(t) \\ dR(t)/dt = \gamma I(t) \end{cases} \qquad (6)$$

4.2 Simulation experiments

- Simulation Parameters

In the following experiments, we will compare SI, SIR and SIPR models to validate the effectiveness of SIPR model. We used the same public parameters for all the three models (Table 2).

Table 2. Parameters used in SI, SIR and SIPR models simulation experiments

Model	I(0)	R[0]	S[0]	Ps[0]
SI	1	unused	1000000	unused
SIR	1	0	1000000	unused
SIPR	1	0	300000-700000	700000-300000
Model	Pi[0]	β	γ	k
SI	unused	0.0000009	unused	unused
SIR	unused	0.0000009	0.05	unused
SIPR	0	0.0000009	0.05	1

- Comparing SI, SIR and IWMM models

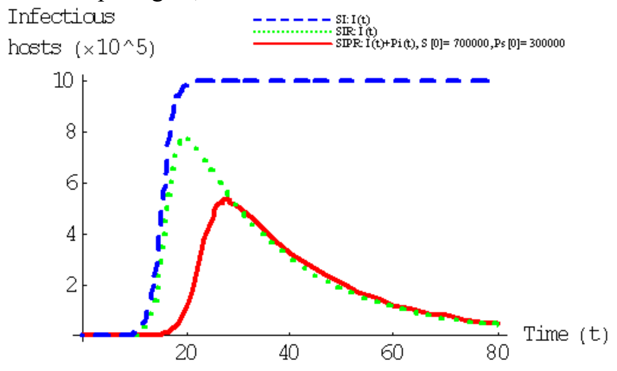


Fig. 4. Comparing SI, SIR and IWMM models

- Comparing different percentage of Ps[0] in SIPR

Intuitively, the more percentage of Ps[0]in N, the more containment ability PoisonWorm will be achieved. Fig.5 shows the number of Infectious hosts for various Ps[0] percentage between 0%, 30%, 50% and 70%. If PoisonWorm occupy more than 50% of the total susceptible hosts, the malicious worm will almost lose its propagation ability.

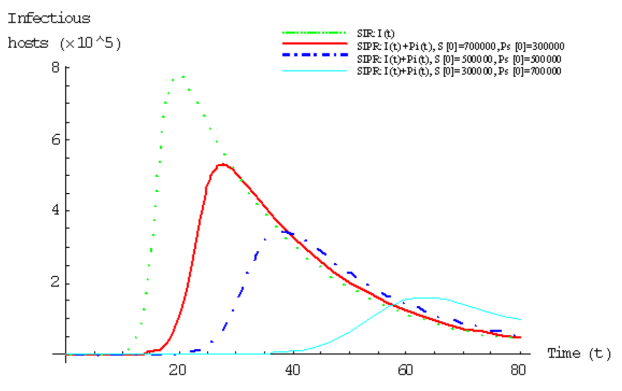


Fig. 5. Different proportion of Ps[0] in SIPR model

- Human intervention comparison of SIP and SIPR models

We can see from Fig.6, though there are fewer infectious hosts in SIPR model, it need less human intervention to remove worms and patch vulnerability.

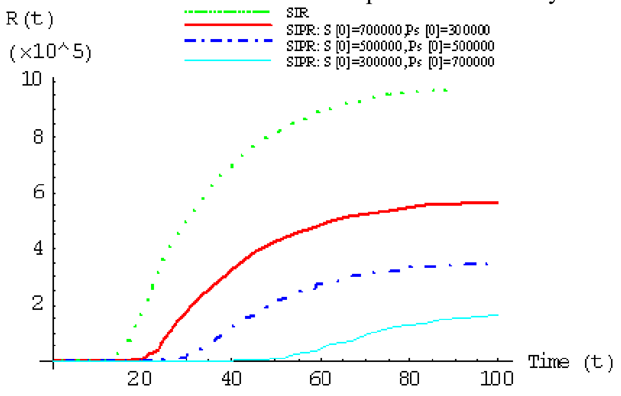


Fig. 6. Human intervention needed comparison in SIR and SIPR model

- Evaluation of extra traffic rised up by PoisonWorm

It is crucial that PoisonWorm would not bring extra traffic burden to the Internet when containing malicious worms. We compute the traffic in SIR and SIPR in a rough way below:

$\text{Flow(SIP)} = \eta I(t) \times \text{ScanPacketSize} + \beta I(t)S(t) \times \text{WormSize}$

$\text{Flow(SIPR)} = \eta[I(t) + Pi(t)] \times \text{ScanPacketSize} + \beta I(t)[S(t) + Ps(t)] \times \text{WormSize} + \beta Pi(t)S(t) \times \text{PoisonWormSize}$

$\text{WormSize} = 100 \times \text{ScanPacketSize}$, $\text{PoisonWormSize} = 1000 \times \text{ScanPacketSize}$, $\eta = 4000$, $\beta = 0.0000009$

In Fig.7, SIPR model produces less and delayed traffic even though the PoisonWorm Size is assumed very large. In reality, the PoisonWorm can be much smaller than the assumed size, a 4k-size basic PoisonWorm was compiled successfully in this paper. So PoisonWorm will not congest the network and is not an addition to the current traffic.

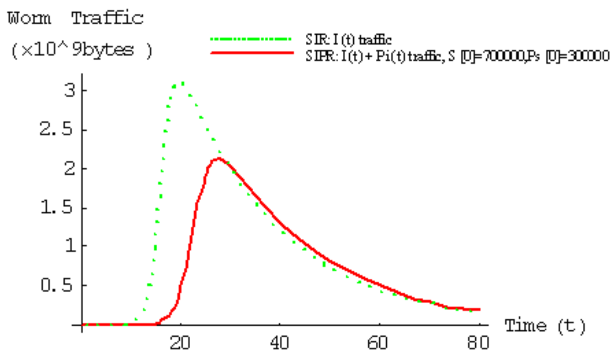


Fig. 7. Worm Traffic comparison in SIR and SIPR model

5. Discussions

Currently, there are so many commercial anti-virus technology and products such as IDS, IPS, AVsoftware, Content-filter Router and Firewall etc. Is PoisonWorm necessary?

5.1 Is PoisonWorm necessary?

We summarize eleven reasons of creating PoisonWorm: It is scalable and inexpensive. When network expands, the PoisonWorm management cost will remain constant. It is different from current worm prevention projects.

PoisonWorm spread with the help of other worms, so the total flow of PoisonWorm and the poisoned worm can be lower than the flow produced by the original worm

PoisonWorm focus on large number of individual vulnerable hosts. It is a complement of security protection software. No matter how excellent AV software is, it cannot protect the hosts without AV software. No matter how useful and timely the patch is, it is nothing for the users without security awareness. But, PoisonWorm is different.

MD5 or public/privacy key authentication, plugin support and botnet-style control methods provide reliable and flexible control ability.

Use P2P architecture, no bottleneck and no single-point failure.

Host&DHT-Based Worm Detection and Signature Verification technology eliminates the disturbance of background traffic greatly. Most current automatic worm detection technology uses the concept of IDS, which is influenced by all kinds of existing worm, scan, spam, and DDoS attack greatly. Every PoisonWorm finds anomaly in local host and then tries to verify the same anomaly with Peer PoisonWorm so the background traffic cannot be influenced.

PoisonWorm can be applied to the research field of worm spread in true Internet environment. Every PoisonWorm has a unique ID, so it is not affected by NAT and dynamic IP. PoisonWorm can log the information of the found worm.

How to control worm is a very difficult issue. Neither patch nor security tools can remove worms if users do not download them. CRII worm still exists even though it broke out early in 2001. PoisonWorm can solve the problem especially.

PoisonWorm is also effective even for those worms which use hit-list, i.e. FlashWorm [4,6,19,20]. FlashWorm makes many worm detection methods useless, such as dark ip, icmp package unreachable and scan rate etc.

High-level worm control is useless for low-level network while low-level network defense costs too much. Host to host path is hard to cut off completely. PoisonWorm can partially solve the problem.

The shared DHT-Based information mechanism gives PoisonWorm a global view.

5.2 Difference between PoisonWorm and anti-worm

The concept of anti-worm (also called good worm) is addressed by Frank [25]. The concept is to transform a malicious worm into an anti-worm which spreads itself using the same mechanism as the original worm and immunizes a host. It resembles PoisonWorm from surface, but they are essentially different:

In short, anti-worm transforms a malicious worm into an anti-worm but PoisonWorm tricks a malicious worm to spread PoisonWorm.

Anti-worm cannot conduct packed worms but PoisonWorm can do easily.

Anti-worm always kills the malicious worms whenever it finds them, while PoisonWorm allows the malicious worm continue to propagate for some time and then kills them.

Anti-worm releases a new worm to kill the existing worm, while PoisonWorm always exists in Internet even if no other malicious worms break out.

The anti-worm itself needs to be generated quickly and spread at least as fast as the original worm. So the active anti-worm is equally disruptive to the network during the spreading process. While PoisonWorm is passive, it replaces the propagation of the original worm; the whole flow of network may even decrease.

Anti-worm aims at buffering overflow worm, including Memory-Residence worm but PoisonWorm aims at all kinds of worm (e-mail/p2p/IM etc) except Memory- Residence worm.

It is hard to produce anti-worm based on multi-vendor worm like Nimda but multi-vendor worm has no impact on PoisonWorm.

Sometimes, it is impossible to produce successful anti-worms for example a worm that needs to negotiate a connection or change the jump address.

Anti-worm needs many resources and widely configuration such as several virtual machines and complex arithmetic but PoisonWorm is only a single program.

6. Future research

6.1 Worm Poisoning technology update and countermeasure

Hooking GetModuleFileName API is only a demonstration of Worm Poisoning concept. VXers can defeat or detour the mechanism easily. They can use many other ways to get its path. Worm Poisoning technology must update its Poisoning technology correspondingly just like Rootkit and anti-Rootkit, Buffer overflow and BOPT (Buffer Overflow Prevention Technology).

6.2 The propagation parameter self-adjusting

After PoisonWorm has successfully propagated for (x) times (in this paper, we simply use x=k=1) using local malicious worm, it will remove the malicious worm. When PoisonWorm infects a new victim, it will not matter whether it carries the original malicious worm (b) and continues to spread for limited (y) times on the new victim. The value of x and y should be zero or a positive integer, and b is a Boolean type value representing whether it carries the original worm or not. The optimized values can be calculated by querying the state of PoisonWorm peers. Optimization can be explained in multiple-ways, including contributing to the least infected hosts, least influence to Internet traffic or earliest to begin to decrease etc.

6.3 Discover suspicious files ability and support plugins

We can't conclude whether a process is malicious or not when it is being scanned. Perhaps this is just a victim of malicious worm which injecting remote thread to its process. To enhance the ability of detecting new attack or achieve new feature, updating new plugins in time is necessary.

6.4 Cooperation with current Worm Containment systems

Current worm defense strategies include filtering infection source, attacked ports and packets with malicious content, anti-worm etc. It is necessary for PoisonWorm to cooperate with them.

6.5 The academic value of PoisonWorm

Giving every PoisonWorm a global-unique id, they will report information of worms found to a control center (though the center has a central architecture, but it is not a part of PoisonWorm defense system. It is not a bottleneck). The collected information can disclose spread process and actual effect to Internet of the unknown worm.

7. CONCLUSION

Worms in the future will be increasingly fast. We must be prepared for the inevitable threat. Outbreaks of the Code-Red, SQL Slammer, MSBlaster, and Sasser worms only reinforce the inadequacy of a system highly dependent on human factors to react accordingly. New defensive mechanisms must be invented to better protect our information systems. We have proposed Worm Poisoning technology in this paper. The reasons for existence of PoisonWorm are illustrated, and the difference between PoisonWorm, Good Worm and antivirus software is clarified. The characteristics are analyzed for Exploiting Worm, E-Mail Worm, Rootkit-based Worm, Memory-Residence Worm, Crack password Worm, IM-based Worm, P2P-based Worm and Packed Worm so on. Common ones are extracted from such the characteristics, which can be utilized by PoisonWorm. It is shown that PoisonWorm can tricks any worms except Memory-Residence Worm and propagate by its self way.

Extension characteristics are analyzed for PoisonWorm applying to worm active defense system. DHT-Based UNKNOWN-WORM Detection and Signature Validation are adopted to improve the coordination way of Earlybird and Autograph systems. And we provide a method based on HASH value searching of dubitable files in this paper, which is a more accurate way to find unknown worms. The techniques developed here would certainly be of interest to other researchers for studying future worms and for inventing new techniques..

References

[1] M. M. Williamson. Throttling viruses: Restricting propagation to defeat malicious mobile code. Technical Report HPL-2002-172, HP Laboratories Bristol, 17 June 2002.

[2] D. Moore, C. Shannon, G. M. Voelker, and S. Savage. Internet quarantine: Requirements for containing selfpropagating code. In Proceedings of the IEEE INFOCOM 2003, March 2003.

[3] N. Weaver, S. Staniford, and V. Paxson. Very fast containment of scanning worms. In Proceedings of 13 USENIX Security Symposium (Security'04), October 2004.

[4] S. Staniford, V. Paxson, and N. Weaver, "How to 0wn the Internet in Your Spare Time" in Proc. of the 11th USENIX Security Symposium (Security'02), 2002.

[5] D. Moore, "The Spread of the Code-Red Worm (CRv2), " http://www.caida.org/analysis/security/code-red/coderedv2 analysis.xml
[6] N.Weaver, "WarholWorms: The Potential for Very Fast Internet Plagues,"http://www.cs.berkeley.edu/_nweaver/warhol.html. August 2001
[7] Zou, W. Gong, and D. Towsley, "Code Red Worm Propagation Modeling and Analysis," in 9th ACM Conference on Computer and Communication Security, Nov 2002.
[8] S. Singh, C. Estan, G. Varghese, and S. Savage. The earlybird system for real-time detection of unknown worms. Paper submitted to HOTNETS-II, August 2003.
[9] C. C. Zou, L. Gao, W. Gong, and D. Towsley. Monitoring and early warning for internet worms. In Proceedings of the10th ACM conference on Computer and communication security,2003.
[10] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford,and N. Weaver. Inside the slammer worm. In IEEE Security and Privacy journal, 2003
[11] Christian Kreibich, Jon Crowcroft. Honeycomb — Creating Intrusion Detection Signatures Using Honeypots, In 2 Workshop on Hot Topics in Networks (Hotnets-II), Cambridge, Massachusetts, Nov. 2003
[12] Hyang-Ah Kim, Brad Karp, Autograph: Toward Automated, DistributedWorm Signature Detection, USENIX Security Symposium, to appear, 2004.
[13] Brent Chun, Jason Lee, Hakim Weatherspoon. Netbait: a Distributed Worm Detection Service. http://netbait.planet-lab.org/berkeley.intel-research.net/bnc/papers/netbait.pdf
[14] David Whyte, Evangelos Kranakis, P.C. van Oorschot. DNS-based Detection of Scanning Worms in an Enterprise Network. Proceedings of the 12th Annual Network and Distributed System Security Symposium, Feb.2005.
[15] 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 Communication Security, November 2002.
[16] Joe Stewart , "Emerging Threats : From Discovery to Protection"
[17] Warlord. Social Zombies: Aspects of Trojan Networks. http://nologin.org
[18] Brandon Wiley, "Curious Yellow: The First Coordinated Worm Design", http://blanu.net/curious yellow.html.
[19] Nicholas Weaver ,Vern Paxson. A Worst-Case Worm. Proceeding of 3rd. Annual Workshop on Economics and Information Security (WEIS04).
[20] S. Staniford, D. Moore,V. Paxson, Nicholas Weaver. The Top Speed of Flash Worms, Proceeding CCS WORM. October 2004.
[21] Jayanthkumar Kannan, Karthik Lakshminarayanan. Implications of Peer-to-Peer Networks on Worm Attacks and Defenses. CS294-4 Project, Fall 2003
[22] Honeynet project ," Know your enemy – Tracking Botnet" http://www.honeynet.org/papers/bots/
[23] Yan Chen, Aaron Beach, Jason Skicewicz.Cyber Disease Monitoring with Distributed Hash Tables: A Global Peer-to-Peer Intrusion Detection System. NWU-CS-04-40 , July 12, 2004
[24] Guofei Gu, Monirul Sharif, Xinzhou Qin, David Dagon, Wenke Lee and George Riley.Worm Detection, EarlyWarning and Response Based on Local Victim Information
[25] Frank C, Emre Can Sezery and Jun Xu. WORM vs. WORM: Preliminary Study of an Active Counter-Attack Mechanism. Proceeding CCS WORM Nov, 2004

Bing Wu received the B.S. and M.S. degrees in Dept. of Architecture from Harbin Institute of Technology in 1992 and 1997, respectively. During 1997-2003, he stayed in Computer Network and Information Center, Harbin Institute of Technology, to manage the HLJ node of CerNet. He now studies in National Computer Information Content Security Key Laboratory of Harbin Institute of Technology for a Ph.D. candidate, focus on anti-virus technology and evaluating internet security situational.