Worm Poisoning Technology and Application

Cui Xiang, Zhou Yonglin, Zou Xin, Wu Bing
National Computer network Emergency Response technical
Team Coordination Center of China(CNCERT/CC)
No.A3 Yumin Road, Chaoyang District, Beijing 100029, China
cuix@cert.org.cn

Abstract

In this paper, the concept of Worm Poisoning and PoisonWorm are presented and the feasibility of Worm Poisoning is emphatically testified. A propagation model called SIRP model and the side-effect to network traffic of PoisonWorm are given and compared to the classical epidemic Kermack-Mckendrick model. We highlight the feasibility and necessity of PoisonWorm and its application in active defense system against Internet worms. Also the technology of P2P-based unknown worm detection and signature verification is briefly introduced.

Keywords

Worm, Worm Poisoning, PoisonWorm, SIPR, Detection, DHT, P2P

1. Introduction

Current strategy against Internet worms is similar to capturing mouse using mousetrap, that is, to clip the occasionally passing mouse and never release until it dies. However, this strategy is less effective than that of spreading pest control chemicals to cause a plague among cockroach group. For infected cockroach, we don't expect it dead at once. We hope it goes back nest and infects others, by which way can kill pests at an exponential rate.

The theory of Worm Poisoning is similar with pest-toxicant production technics. The PoisonWorm functions like the pest-toxicant and the poisoned worm is like the infected pest then.

2. The Concepts of Worm Poisoning and PoisonWorm

Worm Poisoning (also called Worm Spoofing) is a new-invented technology for worm containment. It tricks malicious worms to spread irrelevant file or code by their own mechanisms. The worm which poisons others and propagates by the poisoned worms is called PoisonWorm.

So PoisonWorm is a special worm with active spread motivation, but without self-propagating capability. While it can obtain spread ability

when some other malicious worms break out. It will reduce the negative influence of the malicious worm gradually, and won't cause extra burden to the Internet or its host.

3. The basic principle of PoisonWorm

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.

4. The feasibility of Worm Poisoning technology

There're various kinds and spread mechanism of worms. Whether or not the commonness of worms can be extracted is the issue.

Firstly, PoisonWorm would not carry a signature library like current AV softwares, big size file is not easy to spread. It also does not limit to poison the fast scanning worm(most of unknown worm detection technology particularly aims to fast scanning worm).

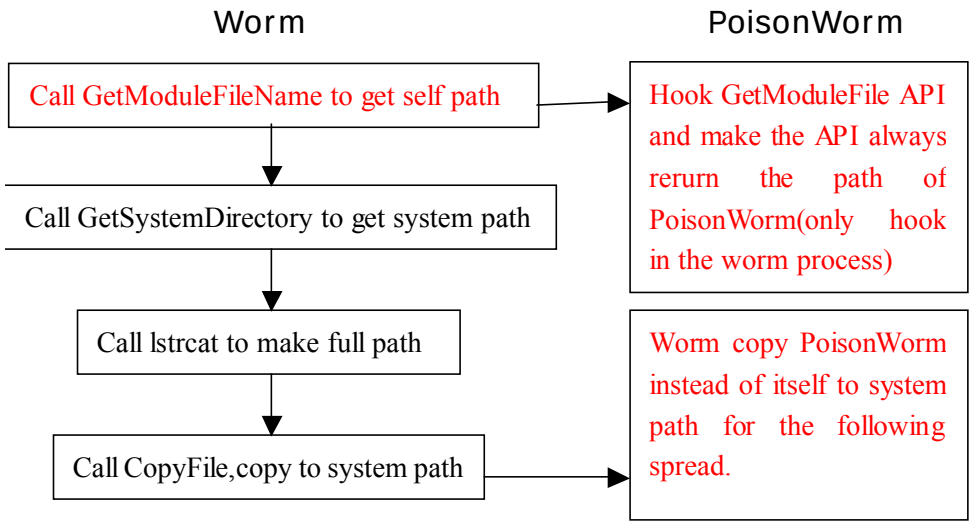
Worm may spread using multi-vectors such as vulnerability、backdoor、e-mail、cracking simple password、IM and P2P etc. There isn't any commonness in the spread mechanism. However, the residence mechanism of worm in the victim host has something in common. The commonness, this paper concerned, is different from the usual anti-virus (which is according to the virus behavior, like API function, executive order and so on) detection technology.

Worm always resides in the compromised victim in order to make itself executed when the OS restarts. It generally follows the steps below:

1). Get its path by calling the 'GetModuleFileName' API;
2). Get system or windows directory path by calling the 'GetSymtemDirectory' or 'GetWindowsDirectory' API;
3). Link system path and worm name to establish the full path by calling 'lstrcat' API;
4). Copy itself to the full path by calling 'CopyFile' API;
5). Modify the Registry or other AutoStart file, which may enable the autorun ability;
6). Begin to spread.

This is the general procedure after worm executed. PoisonWorm makes use of the first step which worm calls GetModuleFileName to get self-path. The purpose of calling GetModuleFileName is to return the path of the running worm process. PoisonWorm hooks the API to always return 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 and then copy PoisonWorm to the destination path and makes PoisonWorm autorun.The most importance is that the malicious worm will spread PoisonWorm instead of itself from now. The following figure will show what PoisonWorm will do:



PoisonWorm is effective to the following type of worms:

3.1 Exploiting Worm

The MSBlaster worm is a typical buffer overflow worm, which broke out at Aug. 2003. its executive procedures are shown below:

1) get self path by calling GetModuleFileName. However, it doesn't try to obtain the system path or copy itself to the system directory.
2) when exploiting remote system successfully, the shellcode binds a cmd shell in the remote system, open tcp 4444 port and listens. MSBlaster worm openes a tftp server in local system(infection source) and listen on udp 69 port.
3) send tftp –i localip GET msblast.exe" string as a cmd to remote tcp 4444 port.
4) remote system runs the tftp command and connects back to the tftp server running in the infection source;
5) the tftp server will transfer the MSBlaster file to the remote system.

Because PoisonWorm has hooked GetModuleFileName API, so MSBlaster will transfer PoisonWorm to remote instead itself. Now we have proved MSBlaster spreads PoisinWorm using its own spread mechanism. Similarly, it works on Nimda、Welchia、Sasser etc, by this mechanism.

3.2 E-Mail worm

For the example of Mydoom and Beagle which broke out in 2004:

1) get self path by calling GetModuleFileName. Copy itself to system path and save the returned path by GetModuleFileName to a global string variable e.g. G;
2) open the file G and encode the file for the purpose of spread as an email attachment;
3) search email addresses, query for the MX record of DNS to get the SMTP Server. Send the email containing the worm attachment to the SMTP Server;

Because PoisonWorm has hooked GetModuleFileName API, so Mydoom and Beagle will encode PoisonWorm and send it to victims via email attachment.

3.3 Worms that using rootkit or cracking password are similar

3.4 IM-based worms are similar to E-Mail Worm.

3.5 P2P-based worms

This type of worms call GetModuleFileName to get self path, then copy itself to the shared directory of P2P software such as Kazaa. Actually, they are more easy than other worms.

3.6 Memory-Residence Worm

This type of worms don't 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.

3.7 Packed worm

Many worms with file carrier are packed using packing software such as UPX. Thus they can not only avoid infected by normal virus but also shrink in size.

PoisonWorm is also effective to packed worm because it dose not infects worms in File, but in memory.

GetModuleFileName is an export function of Kernel32.dll. Kernel32.dll is loaded by OS at the early boot stage (because many system services depend on the functions in Kernel32.dll). Other processes using the API by sharing the loaded Kernel32.dll. The address of the API won't 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 changing of packed worm does no matter.

Now we have proved PoisonWorm can poison all kinds of worms except Memory-Residence worm. The poisoned worms use their own spread mechanism but to spread the file of PoisonWorm, not poisoned worm

itself.

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

The next issue is how PoisonWorm detect the worms to poison?

PoisonWorm is not AV software. So it can only pay attention to wide-spread worms. What PoisonWorm tries to save is not the single infected system but the whole Internet.

For the known wide-spread worms, PoisonWorm recognizes them by filename and path, file or network signature. These signatures and policies are distributed to PoisonWorm via plugins by PoisonWorm Command and Control Center which consists of security expertise and servers.

For the unknown worms(perhaps exist and open to the public for a long time but PoisonWorm doesn't have its signature), PoisonWorm can use existing technology such as AutoGraph[12],EarlyBird[8] 、HoneyComb[11]、NetBait[13]、DSC[24] to discover them. it can also use "DHT-Based UNKNOWN-WORM Detection and Signature Verification" to accelerate the speed of detection. This will be discussed below.

After finding a worm, PoisonWorm terminates the worm process and moves it to a new path. Then it runs the worm process in Suspend mode and use QueueUserAPC to write a piece of "preferential executing" code. When worm process resumes to run, the written code executes firstly. It modifies the address of GetModuleFileName in the worm process in order to make GetModuleFileName always return the path of PoisonWorm. Thanks to the WriteCopy attribute, it won't affect any other normal processes.

Summarizing above, Worm Poisoning is effective for majority worms. CodeRed、Slammer、Witty have no file carrier, but can be controlled and isolated after PoisonWorm finds them.

4. Initialization of PoisonWorm

PoisonWorm can firstly spread using worms captured by honeypots. An unpatched windows 2000 system will be infected by malware in 25 mins on average[16] after connected to Internet. By several honeypots, PoisonWorm can reach many hosts which 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 to find new worms and receive control commands.

5. The extend attribute of PoisonWorm

In order to serve “Worm Active Defence System”, PoisonWorm must add some extended attribute like below:

5.1 Controllable

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 Defence System. So it must be controlled safely. ”MD5+TimeStamp”[17] is an effective way to solve the problem. Every PoisonWorm will carry a hard-coded and same MD5 hash. When receiving a command such as updating signature, it calculates the hash of the command header and compare with its. 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 by sniffer, PoisonWorm checks the TimeStamp each time. Thus one command can and only can be executed for one time. Furthermore, a new MD5 hash is embedded in the command for next time use. PoisonWorm must replace the old MD5 with the new obtained one everytime.

Of course, we can use other way like 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. Furthermore, if private key is missed, it’s very dangerous.

PoisonWorm must support plugin, so it can add new function and remove unnecessary function just like some well-designed bots.

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

PoisonWorm shouldn’t 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 others.

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

1) PoisonWorm uses EarlyBird or DSC to find suspicious flow, uses Autograph to create signature. PoisonWorm also find suspicious file by

heuristic technology (i.e. topology worm can't be detect by EarlyBird style method). 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.

2) 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 Internet, PoisonWorm sends queries to it's peer PoisonWorms. The queried object is the hash of signature or suspicious file. This action is just like P2P software to find particular file name. If most of peer PoisonWorms havn't the object, perhaps it's 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 large. This idea eliminates the influence of background traffic noise greatly.

5.3 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.

6. Is PoisonWorm necessary?

Currently, there're so many commercial anti-virus technology and products such as IDS、IPS、AVsoft、Content-filter Router and Firewall etc. Is PoisonWorm necessary? We summarize ten reasons of creating PoisonWorm:

- It is scalable and inexpensive. When network expands, the PoisonWorm management cost will keep unchanged. It's different with 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(proved in 9.25 below).
- PoisonWorm focus on large number of alone vulnerable hosts. It is a complementarity of security protection software. No matter how excellent AV software is, it can't protect the hosts without AV software. No matter how useful and in time 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 supply 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 use IDS idea, which influenced by all kinds of existing worm、scan、spam、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 can't influence it.

- PoisonWorm can be applied to the research field of worm spread in true Internet environment. Every PoisonWorm has an unique ID, so it's 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 don't download them. CRII worm still exists even though it broke out early in 2001. PoisonWorm can solve the problem in specialty.
- PoisonWorm is also effective even for those worms which use hit-list,ie FlashWorm [4,6,19,20]. Flashworm makes many worm detection method useless, such as dark ip、icmp 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 solve the problem partly.
- DHT-Based information shared mechanism gives PoisonWorm a global view.

7. Difference between PoisonWorm and anti-worm

The concept of anti-worm(also called good worm) is addressed by Frank[25]. The idea 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 looks like PoisonWorm from surface, but they are different in essential:

- In short, anti-worm transform a malicious worm into an anti-worm but PoisonWorm tricks a malicious worm to spread PoisonWorm.
- Anti-worm can't 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 mailicious 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

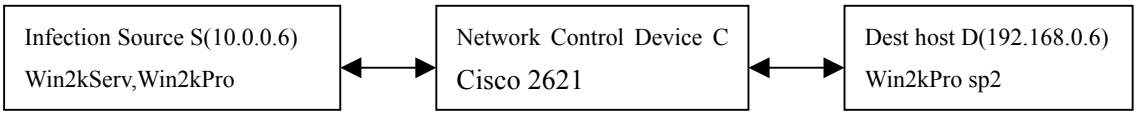
PoisonWorm is passive, it replaces the propagation of the original worm, the whole flow of network may even decrease.

- Anti-worm aims to buffer overflow worm including Memory-Residence worm but PoisonWorm aims to all kinds of worm (e-mail/p2p/IM etc) except Memory-Residence worm.
- It's hard to produce anti-worm based on multi-vendor worm like Nimda but multi-vendor worm has no impact to PoisonWorm.
- Sometimes, it's 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 deploied such as several virtual machines and complex arithmetic but PoisonWorm is only a single program.

8. Experiment result

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

8.2 Experiment 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 infected by worms in S quickly and traffic controll. Because the scan policy of worms is different, D need some time to be scanned successfully. To solve the time delay problem, we configured a DNAT (Destination Network Address Translation) 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, and 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 transparently.

To avoid D 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. After several seconds of 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 is 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 very fast and reliably.

8.3 Testing 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 running steady. 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 is shown in figure 1. Tested sasser worm,we got a similar successful result like MSBlaster shown in figure 2.The two figures show what the two worms actually transferred in the network:not themselves, but PoisonWorm.

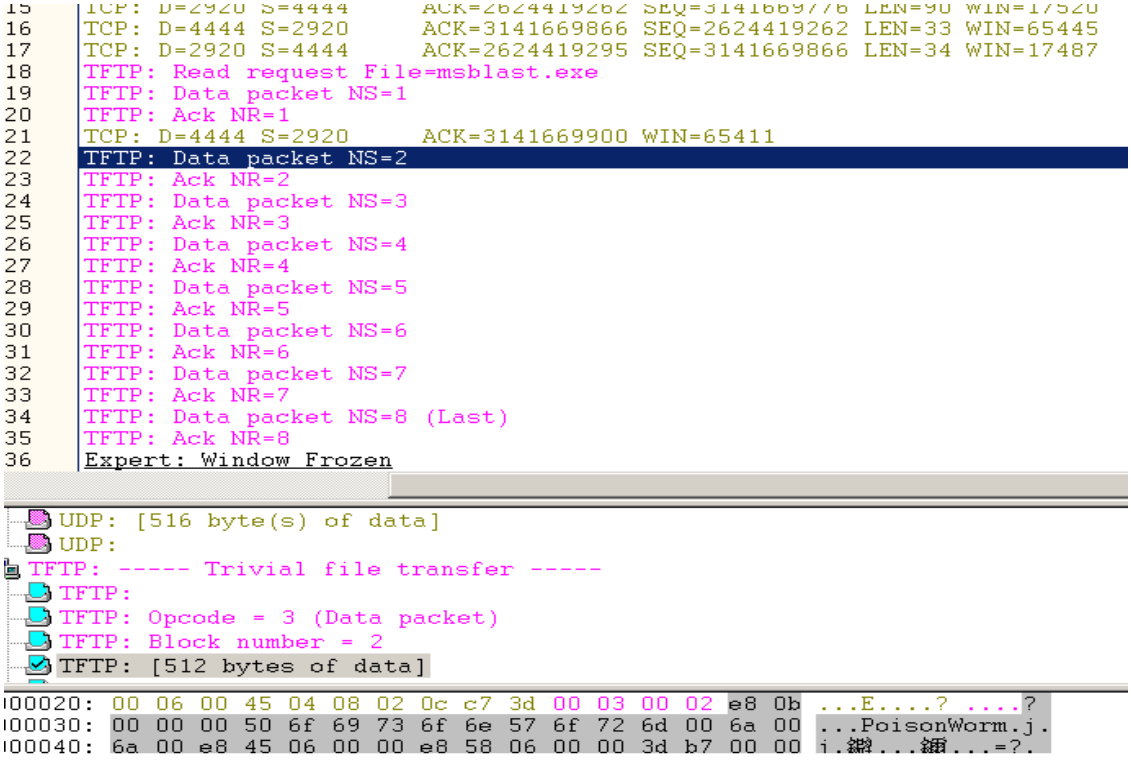


Figure 1: packets sent by MSBlaster worm

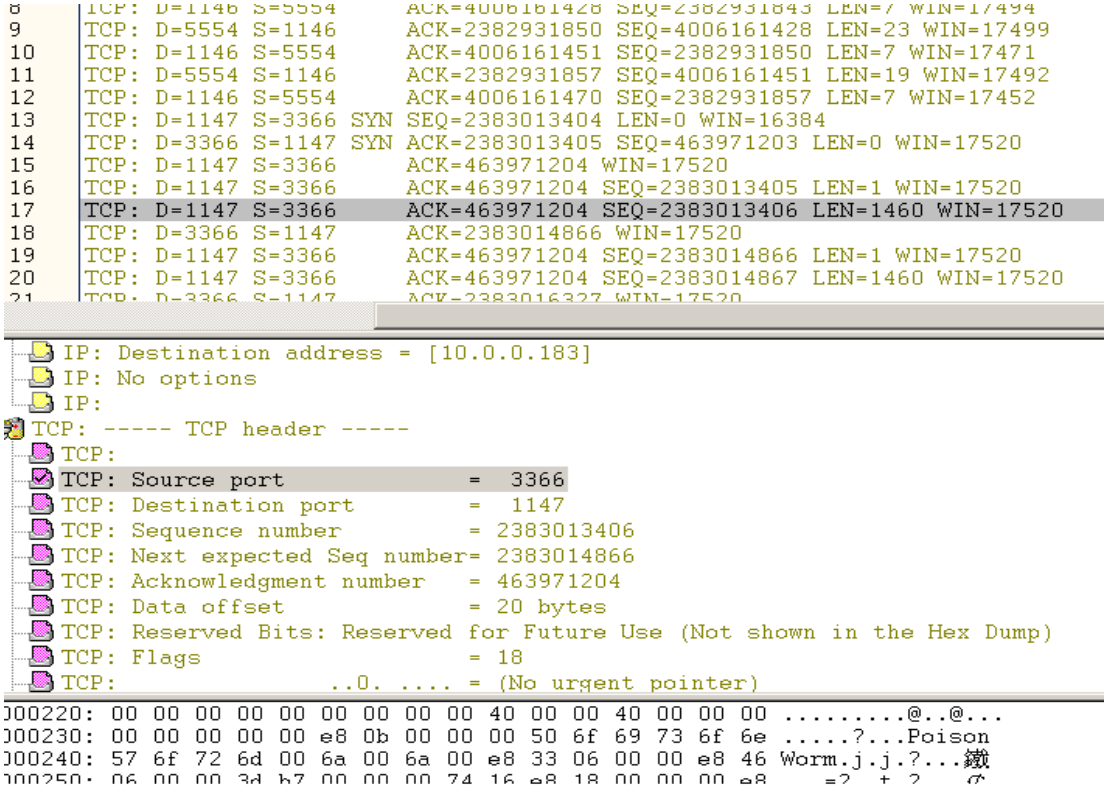


Figure 2: packets sent by sasser worm

For Welchia(MSBlaster-remover) and NetSky worm, we only tested in local host. Executing PoisonWorm after executing Welchia and NetSky, we observed 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 actually.

9. SIPR Model analysis

9.1 SIPR Model introduction

The SIPR (susceptible-infected-poisoned-recovered) model is presented here in order to analyze Worm Poisoning Technology. It's an epidemic model that assumes each host to exist in one of 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 meaning of the notations are explained in fig.3.

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)

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

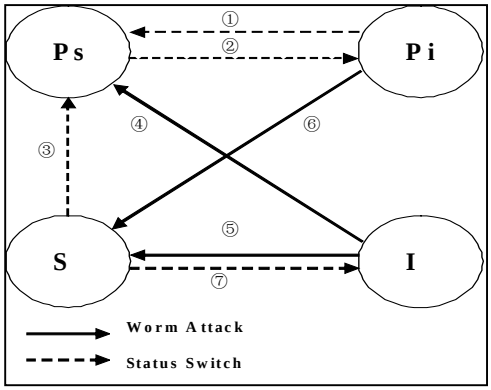


Figure 3: SIPR model attacking and status switching graph

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:

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

9.2 Simulation experiments

9.2.1 simulation parameters

In the following experiments, we'll 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]	Pi[0]	β	γ	k
SI	1	unused	1000000	unused	unused	0.0000009	unused	unused
SIR	1	0	1000000	unused	unused	0.0000009	0.05	unused
SIPR	1	0	300000-700000	700000-300000	0	0.0000009	0.05	1

9.2.2 Comparing SI, SIR and SIPR models

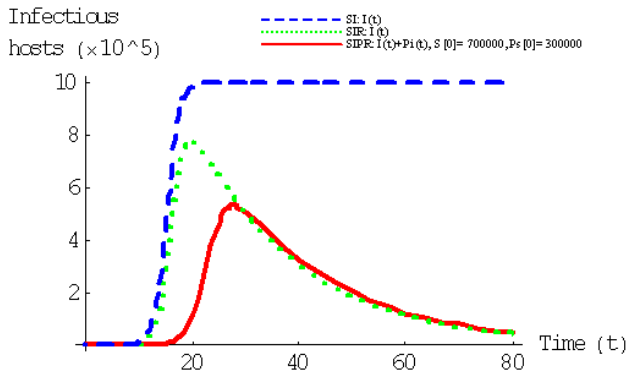


Figure 4: Comparing SI, SIR and IWMM models

9.2.3 Comparing different percentage of Ps[0] in SIPR

Intuitively, the more percentageof Ps[0]in N, the more containment ability PoisonWorm will achieve. 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.

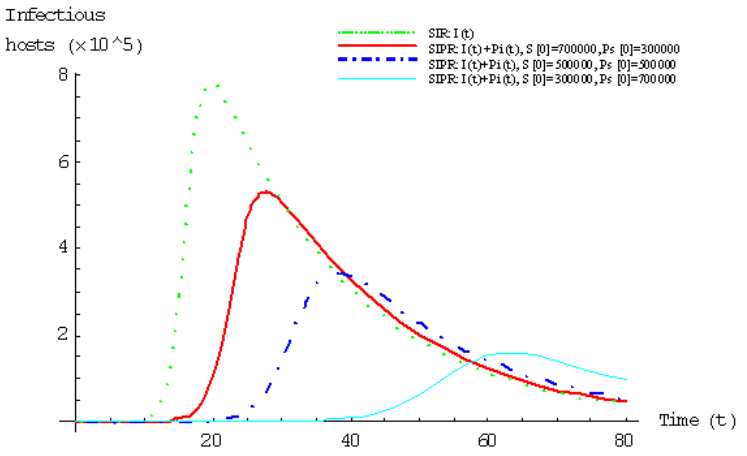


Figure 5: Different proportion of Ps[0] in SIPR model

9.2.4 Human intervention comparison of SIP and SIPR models

We can see from Fig.6, though there're fewer infectious hosts in SIPR

model, it need less human intervention that remove worms and patch vulnerability.

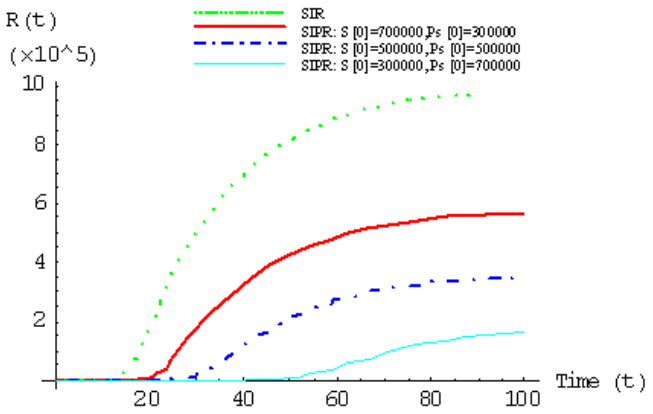


Figure 6: Human intervention needed comparison in SIR and SIPR model

9.2.5 Evaluation of extra traffic rised up by PoisonWorm

It's crucial that PoisonWorm won't bring extra traffic burden to the Internet at the same time of containing malicious worms. We compute the traffic in SIR and SIPR in a rough way below:

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

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

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

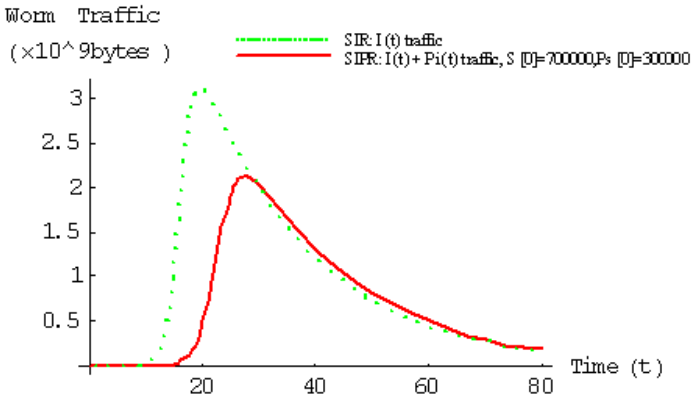


Figure 7: Worm Traffic comparison in SIR and SIPR model

We can see in Fig.7, SIPR model produces less and delayed traffic even though the PoisonWormSize is assumed very large. In reality, the PoisonWorm can be much smaller than this assumed size, a 4k-size basic PoisonWorm was compiled successfully in this paper. So PoisonWorm won't congest the network and isn't an addtion to the current traffic .

10. Related research

The concept of Worm Poisoning is firstly addressed in the paper. Worm

similar to PoisonWorm is not discovered in the wild up to date (annotation: the author had implemented a PoisonWorm based on Worm Poisoning technology and tested it successfully under Win2kpro sp4, Win2kServ sp3 and WinXP sp2).

No related research of verifing the signature and suspicious files using P2P technique is discovered. (PoisonWorm hashes the signature or file as the querying object). There's P2P-based IDS [23] in abroad, but it's very different from the technology in the paper.

In the research field of automatic unknown scanning worm detection and signature creation, there're 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're 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]、Anti-wormetc. Most of these control mechanism must deploy hard and soft equipments widely which costs a lot. Thanks to Internet designed to have very strong connectedness, it's hard to cut down all spread path.

10. Future research

10.1 Worm Poisoning technology update and countermeasure

Hooking GetModuleFileName API is only a demonstration of Worm Poisoning idea. 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).

10.2 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 should remove the malicious worm. When PoisonWorm infects a new victim, whether or not 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 or not it carries the original worm. The optimized values can be calculated by querying the state of PoisonWorm peers. Optimization can be explained in multi-ways including contributing to the least infected hosts、least influence to Internet trffic or earliest to begin to decrease etc.

10.3 Discover suspicious files ability and support plugins

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

10.4 Cooperation with current Worm Containment systems

Current worm defense strategies include filtering infection source,attacked ports and packets with malicious content, anti-worm etc. PoisonWorm should cooperate with them.

10.5 The academic value of PoisonWorm

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

10.6 How to defend worms which spread using Worm Poisoning technology?

11. ACKNOWLEDGMENTS

We would like to thank PhD Yuejin Du from CNCERT/CC for his excellent proposal and guidance. We also thank Liu Yang from CNCERT/CC for his help on translation.

12. 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 techniques developed here would certainly be interesting to other researchers for studying future worms and for inventing new techniques.

13. 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.
[7] C. C. 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, www.csd.uch.gr/~hy558/papers/honeycomb.pdf
[12] Hyang-Ah Kim ,Brad Karp.Autograph: Toward Automated, DistributedWorm Signature Detection,www-2.cs.cmu.edu/~hakim/ autograph/autograph-usenixsec2004.pdf
[13]Brent N. Chun..Netbait: a Distributed Worm Detection Service 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
[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 .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
[20] Stuart Staniford,,David Moore,Vern Paxson,Nicholas Weaver.The Top Speed of Flash Worms
[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”
[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 Casta.neda¤y, Emre Can Sezery and Jun Xu. WORM vs. WORM: Preliminary Study of an Active Counter-Attack Mechanism. http://www.icir.org/vern/worm04/castaneda.pdf