COLLABORATIVE DETECTION OF POLYMORPHIC WORMS

S.Priyadharshini, (M.Tech), P. Sunil Kumar, B.Tech., M.E

Periyar Maniammai University

Vallam, Thanjavur

ABSTRACT- Internet worms pose a serious threat to computer security. Traditional approaches using signatures to detect worms pose little danger to the zero day attacks. The focus of malware research is shifting from using signature patterns to identifying the malicious behavior displayed by the malwares. This project presents a novel idea of extracting variable length instruction sequences that can identify worms from clean programs using process monitoring and intrusion detection techniques. The analysis is facilitated by the program control flow information contained in the instruction sequences. Based upon general statistics gathered from these instruction sequences this proposed work formulates the problem as a binary classification problem and built tree based classifiers including decision tree, bagging and random forest.

Keyword: worms, zero day attacks, malwares

INTRODUCTION

We live in the information age and it is very useful for us to share and store information conveniently for the purpose of education, research, increase in productivity and quality of service, electronic storage and easy retrieval etc. One efficient and cost effective way of ensuring these is through the use of computer networks. A network is a connected collection of devices and end systems such as computers, servers, printers which can communicate with each other and share resources. Networks are implemented and in use in homes, small offices, large enterprise and in governmental organizations. Their components include personal computers, interconnections, switches, router, firewalls, etc. Each of these devices perform distinct functions to enable information storage or access information in a faster, reliable, secured, cost effective and convenient manner. The internet, which is a 'network of networks,' has been growing rapidly because it directly affects business, education, governmental activities and social interaction. Because of the benefits of using the internet, a lot of valuable and sensitive information travel through the network and these data transfer attracts attackers to steal, intercept or destroy valuable information and to disrupt normal network connection for fun, fame or money. One way of deploying such attacks is through the deployment of computer worms.

A computer worm is malicious software (malware) designed to attack a network with the aim of either stealing information on the network, destroying network resources or to deny legitimate network users of needed resources- thistype of denial attack is called denial of service (DoS). It is a self-replicating special type of virus program that propagates itself via network connections, taking advantage of security flaws in computers on the network. Worms do not need human intervention to propagate. Worms pose a big threat to network because they evade network security measures stealthy, that is, unnoticed. A computer worm, after it is released, includes the following phases: target finding, worm transferring, worm activation, and worm infection. During the first two phases, the worm is active over the internet thus making it possible to be detected by network-based intrusion detection systems (NIDS). The other two phases are hard to detect by

NIDS. The first step of a worm's life is to find targets.

A worm may find its target or next victim using many different strategies. These strategies include: blind scan- the blind scan method includes sequential, random and permutation scanning, though they are probabilistic because of high failure connection rate and because the worm has no prior knowledge about the targets. The second target finding strategy is using a hit-list. The hit-list is a list of pre-scanned vulnerable addresses, by this the worm knows exactly where the targets are. The variation for this type of target finding strategy is that the larger the size of the hit-list, the more accurate and the more damage it can cause. Thirdly, the use of network topology can enable a worm to find its target because many hosts on the internet store information about other hosts on the network revealing their vulnerabilities. Fourthly, a passive strategy is another approach worms employ in finding targets by patiently waiting for victims to communicate with where the worm is resident. Lastly, web searching is another strategy used by worms to find their targets because web searches avoid being detected by traditional detection techniques.

The second phase of a worm's life cycle is its transferring or propagation. It does this by employing three different schemes, namely: self-carried- this method allows the worm code to be transferred in a packet by itself. Second channel-this method allows the worm, after finding its target, to go into the target and download the worms code through a

'backdoor' that has been installed by some applications. Embedded- this method allows the worm to attach its code to legitimate traffic for example an e-mail in order to hide itself. This method is very deceitful and often unnoticed. The third phase of a worm's life cycle is worm activation, that is, how the worm is transmitted over the network. There are two basic ways in which worms are transmitted over the network; they are transmission control protocol (TCP) and user datagram protocol (UDP). The main difference is that TCP worms are connection oriented because they require a connection to be established before infection can begin, unlike UDP worms that are connectionless and requires no connection to infect targets, this makes them spread very rapidly.

The last phase of a worm's life cycle is worm infection. This phase of the worm is associated with the actual worm code format. Worms usually send their code in a direct manner which causes detection systems to identify them quickly. Worms can be monomorphic in format; filling the code with irrelevant data but maintaining a single signature. They can also be polymorphic in format; that is, their code changes dynamically by scrambling them so that the worm takes different forms from different views though maintaining the same. The last phase of a worm's life cycle is worm infection. This phase of the worm is associated with the actual worm code format. Worms usually send their code in a direct manner which causes detection systems to identify them quickly. Worms can be monomorphic in format; filling the code with irrelevant data but maintaining a single signature. They can also be polymorphic in format; that is, their code changes dynamically by scrambling them so that the worm takes different forms from different views though maintaining the same function.

This type of worm format is very hard to be detected by signature-based detection. Another worm format is metamorphic worms. It changes not just appearance but also behavior. Worm structures may have features that enable them to locate targets, propagate infections, a remote control that enables the author to control the worm remotely, an update interface that enables author to update the worm's code. Some examples of worms are Stuxnet, Morris Worm, Code Red, Nimda, Slammer, Sasser, Witty, etc.e function. This type of worm format is very hard to be detected by signature-based detection. Another worm format is metamorphic worms. It changes not just appearance but also behavior. Worm structures may have features that enable them to locate targets, propagate infections, a remote control that enables the author to control the worm remotely, an update interface that enables author to update the worm's code. Some examples of worms are Stuxnet, Morris Worm, Code Red, Nimda, Slammer, Sasser, Witty, etc.

Detecting worm attacks has become a thing of concern because of the kind of havoc they can cause on networks. This research focuses on analyzing how intrusion detection systems (IDS) detect worm attacks. An intrusion detection system is hardware or software that is installed on the network or host computers that monitor data traffic on the network in order to discover illegitimate or malicious traffic that disobeys the security policy of a particular network. IDS can be network-based or host-based. Network-based intrusion detection systems (NIDS) analyze network traffic at all layers of the open system interconnection (OSI) model and check for anomalous packet behavior or unwanted packet signatures, and when these are detected, it raises an alarm, calling for the attention of a security administrator. They are easy to deploy and can monitor traffic from many systems at once. Host-based intrusion detection systems (HIDS) are usually software installed on host systems and they generally analyze network traffic and system specific settings such as local security policy, local log audits, and so on. Although both NIDS and HIDS have their strengths and limitations, a use of both at different points in the same network improve the effectiveness of threat detection.

IDSs uses two schemes in detecting illegitimate traffic, these schemes are signature-based and anomaly-based detection methods. It is not just enough to detect worms. The next thing that should be done is to contain them and trace them back to the source. This research will only analyze the approaches of tracing back attacks. Attack trace back just as the name implies, is merely the tracing of attacks back to their origin. This can be seen as a sort of network forensics, which is the capture, recording and analysis of network events in order to discover the source of security attacks or other problem incidents. It is a reactive measure of attack trace back. There are many reactive approaches for attack trace back, they include; link testing, logging, ICMP (internet control message protocol) trace back and packet marking.

PROBLEM STATEMENT

In order to detect an unknown (zero-day) worm, a straightforward way is to use various threshold-based anomaly detection methods. We can directly use some well-studied methods established in the anomaly intrusion detection area. However, many threshold-based anomaly detections have the trouble in dealing with their high false alarm rate. In this paper, we do not try to propose another threshold-based anomaly detection method. Instead, we present a non-threshold based detection methodology, “trend detection”, by using the principle “detecting monitored traffic trend, not burst”. Traditional threshold-based anomaly detection methods try to detect a worm by detecting either the long-term or the short-term burst of monitored traffic. However, the monitored data contains noisy background traffic that is caused by many other factors besides the worm we want to detect, such as some old worms’ scans or hackers’ port scans. Thus traditional threshold-based detections usually will generate excessive false alarms. In the case of worm detection, we find that we can take advantage of the difference between a worm’s propagation and a hacker’s intrusion attack: a worm code exhibits simple attack behaviors and its propagation usually follows some dynamic models because of its large scale infection; on the other hand, a hacker’s intrusion attack, which is more complicated, usually targets one or a set of specific computers and does not follow any well-defined dynamic model in most cases.

EXISTING SYSTEM

Internet attacks such as Distributed Denial-of-Service (DDoS) attacks and worm attacks are increasing in severity and frequency. Identifying and mitigating realtime attacks is an important and challenging task for network administrators. An infected host can make a large number of connections to distinct destinations during a short time. Such a host is called a superpoint. Detecting superpoints can be utilized for traffic engineering and anomaly detection. The existing sysem proposes a novel data streaming method for detecting superpoints and proves guarantees on its accuracy with low memory requirements. The superior performance of this method comes from a new data structure, called Vector Bloom Filter (VBF), which is a variant of standard Bloom Filter (BF). The VBF consists of six hash functions, four of which take some consecutive bits from the input string as the corresponding value, respectively. The information of super-points is obtained by using the overlapping of hash bit strings of the VBF. Theoretical analysis and experimental results show that the proposed method can detect superpoints precisely and efficiently through comparison with other existing approaches.

Disadvantages

I. Existing methodology is not suitable to the recent day trends of worm attacks.

II. The result will not be accurate if the worm is from different sources.

III. Worm detection only implemented but prevention and security is not implemented in the existing methodology.

PROPOSED SYSTEM

In out proposed system an effective algorithm named **Collaborative Internet Worm Detecton (CIWD)** is used for early detection of the presence of a worm and the corresponding monitoring system. Based on epidemic model and observation data from the monitoring system, by using the idea of

"detecting the trend, not the rate" of monitored illegitimated scan traffic, the system proposes to use a Malware filter to detect a worm's propagation at its early stage in real-time. In addition, the system can effectively predict the overall vulnerable population size, and correct the bias in the observed number of infected hosts. Our simulation experiments for Code Red and SQL Slammer show that with observation data from a small fraction of IP addresses, the system can detect the presence of a worm when it infects only 1% to 2% of the vulnerable computers on the Internet.

Advantages

Fig. 1. The proposed system is modified to support the worm detection mechanism of current trends The CIWD shows better performance than the existing systems and effective in detecting internet worms.

Fig. 2. The proposed system not only detects the internet worms it also provide the prevention security for the server data.

PROPOSED ARCHITECTURE

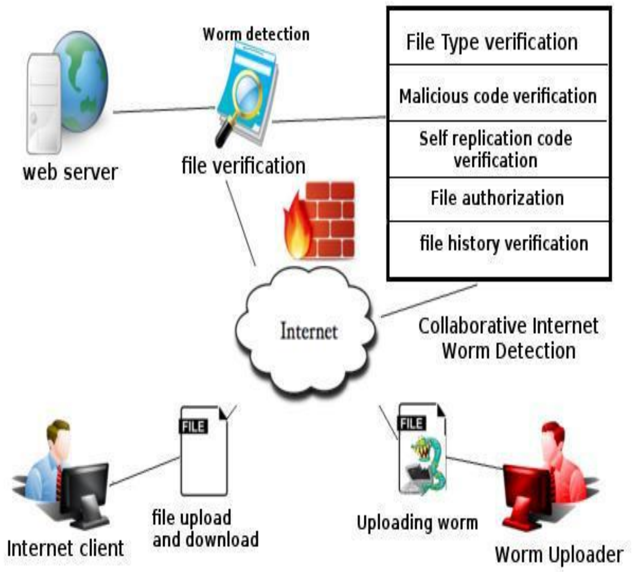


Fig – Architecture diagram of Collaborative Interdependent Worm Detection

MODULES

The proposed Collaborative Internet Worm Detection algorithm uses the following modules in order to verify the uploading and downloading files on the network.

[1] File Type verification

[2] Malicious Code Verification

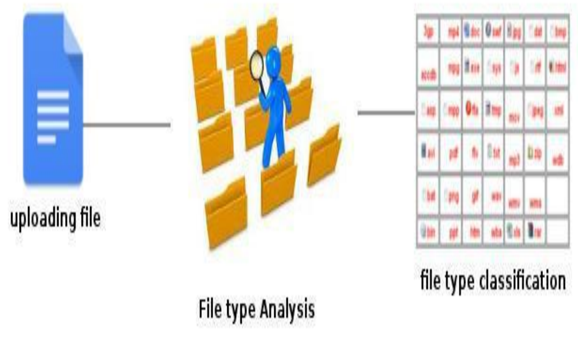
[3] Self Replication Code Verification

[4] File Authorization

[5] File History Verification

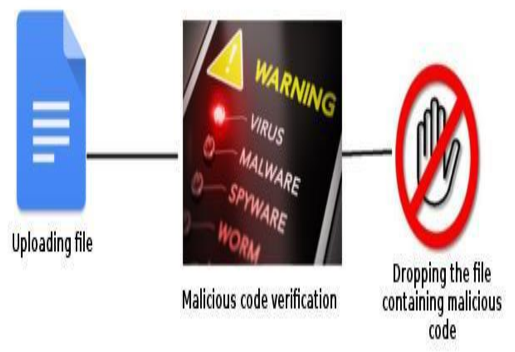
File Type Verification

This module is implemented for file type verification, you can process files based on their true file type, so you can take more precautions with risky file types like EXEs, perhaps setting different policies or rules based on file type. Spoofed file types indicate potentially malicious intent, so to mitigate this risk, Metadefender Core can block files with incorrect extensions, preventing for instance EXE files posing as TXT files from entering your organization. Every file has a particular format. Files are either binary or ascii. Common ascii files would be simple text or more complicated formatted text such as PDF or XML



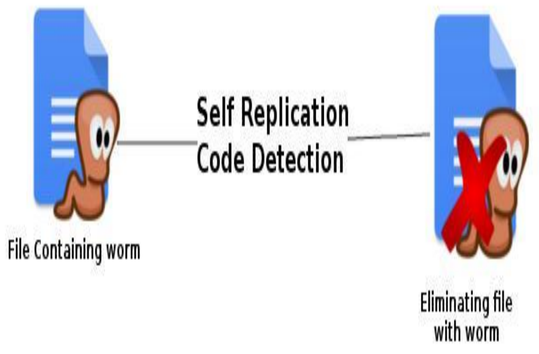
Common binary files are images (jpeg, gif, png) or compressed files. But the file formats can be layered, such as DOCX or PPTX. These are a collection of ascii XML files in a zip archive which makes it a binary file.

Malicious code verification


Uploading file
WARNING
VIRUS
MALWARE
SPYWARE
WORM
Malicious code verification
Dropping the file containing malicious code

Perhaps the first large-scale practical instance of the language-based approach was the Java programming language. Javas language-based mechanism is designed to protect against malicious applets. The Java runtime environment contains a bytecode verifier that is supposed to ensure the basic properties of memory, control flow, and type safety. In addition, a trusted security manager enforces higher-level safety policies such as restricted disk I/O. The Java compiler produces platform-independent virtual machine instructions or bytecode that can be verified by the consumer before execution. The bytecode is then either interpreted by a Java virtual machine (VM) interpreter or further compiled down to native code. Early versions of Java contained a number of highly publicized security flaws. For example, a subtle defect in the Java type system allowed an applet to create and control a partially instantiated class loader. The applet could then use this class loader to load, say, a malicious security manager that would permit unlimited disk access.

Self Replication Code Verification



An approach to the detection of malicious software by detecting its ability to self-replicate is proposed, implemented and tested. The approach is justified by the following realities most malicious programs propagate themselves through the Internet to maximize the impact of the information attack; self-replication of legitimate programs is quite uncommon; number of practical self-replication techniques is quite limited and is to be repeatedly used by new malicious codes. A Source Code Analyzer operating as a specialized compiler (interpreter) and a special syntax library were developed for the detection of self-replication functionality in source codes/scripts prior to execution. Major building blocks of the existing self-replication techniques were defined in the domain of system calls and their attributes, and a procedure for the reconstruction of these blocks by analyzing the flow of system call was established. A dynamic Code Analyzer and System Calls Monitor were developed for the run-time detection of the attempted self-replication in executable and encrypted executable codes. The efficiency of the developed technology, including the ability to detect previously unknown malicious programs has been experimentally demonstrated.

File Authorization

Access control in a public cloud is usually accompanied by safety certification. Users have to login to the cloud platform to access data by using the certificate interface provided by the service providers and manage the resources accordingly. In a hybrid cloud, access right permissions usually encounter problems such as inconsistency of file access rights which may cause the employees being unable to access the same resources on the public cloud. File authorization is performed by the File Authorization Module. It checks the access control list (ACL) of the server file handler to determine whether a user should have access to the file. ACL permissions are verified for the file identity and accessible security.

File History Analysis

File History analysis tools are designed to analyze source code and/or compiled version of code in order to help find security flaws. Ideally, such tools would automatically find security flaws with such a high degree of confidence that what's found is indeed a flaw However, this is beyond the state of the art for many types of application security flaws. Thus, such tools frequently serve as aids for an analyst to help them zero in on security relevant portions of code so they can find flaws more efficiently, rather than a tool that just automatically finds flaws. Some tools are starting to move into the IDE. For the types of problems that can be detected during the software development phase itself, this is a powerful phase within the development life cycle to employ such tools, as it provides immediate feedback to the developer on issues they might be introducing into the code during code development itself. This immediate feedback is very useful, especially when compared to finding vulnerabilities much later in the development cycle.

Scope of Defense

Different locations of implementation give different scopes of worm detection and containment. Figure 6 illustrates the correlation of these different network levels. Most IDSs are designed for local area or enterprise network detection and containment. The local area or enterprise network is a clearly defined entity and is normally controlled by one single organization, which has central management when it comes to deciding the type of IDS to implement.

A broader extent that is well defined, is the scope of an ISP or AS, which has multiple customer networks connected to it. After detecting worms from a certain customer network, the ISP can slow down or block off partial traffic from that network to prevent worms from spreading to other customer networks. The detection might be more complex for signature- based algorithms because of dealing with large amounts of traffic, so anomaly-based algorithms may be more feasible. Wagner and Plattner proposed an entropy2-based anomaly detection system to detect worms in fast IP networks (networks with large amounts of traffic) such as the Internet backbones . Essentially, the larger the coverage, the more accurate the normal model definition. The DAW architecture is another example of a system for this scope, which is implemented with anomaly-based detection and containment inside an ISP network. Different parameters are used for different scopes of detection. As the scope grows bigger, detection may be rougher, but containment is more effective. Worms spread at very fast speeds. The damage of a worm outbreak is normally very broad, often across countries. In previous sections we have seen that many of the worm detection algorithms are implemented based on monitoring larger size networks. Also showed that worm containment is only practical if a large fraction of the Internet unites and works together. This leads us to conclude that global scope is necessary in defending against worms proposed a system combining both control plane data (routing data) as well as data plane data (packet headers and payloads) to detect and contain Internet worms mor efectively. In this system, anomalies detected on the data plane are used to identify ASs that are associated with the attacks and apply control plane filters to contain them. Furthermore, anomalies detected on the control plane (e.g., IP hijacking) can be used to deploy strict data plane controls on a particular

Conclusion

Identified the characteristics of existing and hypothetical forms during the target finding and propagation phases of a worm’s life cycle. They are classified based on target finding, propagation, transmission scheme, and payload format. Current detection algorithms are organized based on the categories of signature-based, anomaly-based, or hybrid. Evaluated these categories against worm characteristics. Classified current containment schemes based on the methods they use to control the spread of worms. Aalso explored the implementations of detection and containment at different network locations and system scopes. An ideal system should use a combination of schemes to have more comprehensive coverage. Different detection schemes are useful at different levels of implementation. So far, there is no ultimate solution to deal with all existing and hypothetical worms. New attack technologies are being developed every day, and the threat constantly exists. We have pointed out the remaining challenges and future work to be done based on the analysis of current algorithms. So far, there are limited solutions for detecting passive and topological scanning worms, flash worms, and metamorphic worms; nevertheless, as pointed out in research by Kienzle and Elder the majority of new worms coming out every day are not novel and are derivative in nature. As a result, by defending against yesterday’s worms, we can effectively protect ourselves against most new worms; at the same time, we also need to prepare for the threats of new novel worms that can hit us in the future.

REFERENCES:

[1] Ratinder Kaur and Maninder Singh "A Surveyon Zero-Day Polymorphic Worm Detection Techniques" Department of Computer Science and Engineering, Thapar University, Patiala, India. IEEE 2014.

1. Pele li, Mehdi Salour, and Xiao Su, San Jose state university "A survey of internet worm detection and containment" 1st Quarter 2008, volume 10, no. 1.

2. Vishrut Sharma ,Member of ACM, IEEE "An Analytical Survey of Recent Worm Attacks" IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.11, November 2011

3. Per-Flow Traffic Measurement Through Randomized Counter Sharing, Tao Li, Shigang Chen, and Yibei Ling, OCTOBER 2012

4. Robust Network Traffic Classification, Jun Zhang, 2013

5. Streaming Solutions for Fine-Grained Network Traffic Measurements and Analysis, Faisal Khan, Nicholas Hosein, Soheil Ghiasi, APRIL 2014

6. Cardinality Change-based Early Detection of Large-scale Cyber-Attacks, Wenji Chen and Yang Liu and Yong Guan, 2013

7. Behavior Analysis of Internet Traffic via Bipartite Graphs and One-Mode Projections, Kuai Xu, Member, IEEE, ACM, Feng Wang, Member, IEEE, and Lin Gu, Member, IEEE, JUNE 2014

8. H. Zeidanloo, M. Shooshtari, P. Amoli, M. Safari, and M. Zamani, "A taxonomy of botnet detection techniques," in Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, vol. 2, july 2010, pp. 158 – 162.

9. A. Ramachandran, N. Feamster, and G. Tech, "Understanding the network-level behavior of spammers," in Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications (2006), 2006, pp. 291–302.