A Novel Approach to Worm Detection Systems

Yazed B. Al-Saawy, Antonio Cau and François Siewe
Software Technology Research Laboratory
De Montfort University
The Gateway, Leicester LE1 9BH, UK
Email: {yazed,acau,fsiewe}@dmu.ac.uk

***Abstract*—Computer worms are a type of malicious malware that prey on networked machines. A number of different detection mechanisms have been presented in the literature to detect worms. However, a common drawback of these mechanisms is that any failure to detect the worms results in damaging the real machines. This study proposes a new approach to detection that goes beyond the currently available signature and behavior-based approaches. In contrast to the traditional worm detection system (WDS) that use signature and behavior-based approaches, our proposed approach is based on detection by the damage caused by worms on dummy machines rather than the real machines. The proposed WDS adds additional security as compared to the currently used systems by allowing worms to conduct their normal behavior in a dummy host, thus protecting the rest of the network from damage. The proposed WDS was designed within a network setting and was capable of sending and receiving files and messages between hosts as part of the overall detection mechanism.**

***Keywords*—Worms detection; computer security; dummy host; behaviour-based; signature-based; damage**

I. Introduction

Computer worms impose a threat because they prey on networked machines which are increasing becoming the backbone of any organisations. The damage that occurs in these networks can be costly and may bring organisations down. There have been responses to these threats and the main approach to resolve this problem is to detect worms. There are mainly two approaches to detect worms namely; signature-based detection and behavior-based detection. However, there are certain limitations with both approaches. As an example, the signature-based detection depends on a database consisting of the signatures of known worms which means that this approach can detect only known worms and any new worms will remain undetected. A response to this limitation was behavior-based detection whereby detection is achieved through monitoring changes in the behavior which could indicate the presence of a worm. Although this approach overcomes the limitations of signature-based detection, however, the detection often occurs after the infection and damage has already taken place which is too late.

Therefore, there is a need for an approach to detect worms that answers the limitations of the two existing approaches. The proposed worm WDS is based on the behavior-based detection technique but allows the worms to carry out their behavior and subsequently cause damage in a safe environment within which detection takes place.

The proposed WDS detects worms through detecting the replication behavior of worms as well as any potential damage caused. The worm detection takes place in a dummy host which is a real machine located within the hub of a network that allows worms to replicate and cause damage. In addition to the behavior-based detection, signature-based detection is also included in the proposed WDS to detect known worms. Thus the unique contribution of this study is detection of worms through behavior and damage in a dummy host.

II. Related work

Threats from malware are a significant problem in the world of computer security [1] and because these threats are growing the signature-based and behavior-based detection techniques are increasingly becoming obsolete [2]. The main reason for anti-malware technology becoming obsolete is because it is mostly reactive [3].

A. Signature-based detection

Signature-based detection uses a database of signatures or byte sequences of known malware which are used to determine if a file or message is infected by checking these sequence of bytes against the database ([4] ,[5]) using a detection algorithm [6]. One of the main limitations of this approach is that it depends on other machines being previously infected in order to derive signatures ([7], [8]). Therefore, a clear limitation of this approach is that it cannot detect unknown malware because signature is not known.

Moreover, even though it is claimed that this technique can detect malware with known signatures, there is a wide variety of possible variants of these signatures they are difficult to detect [9]. A problem associated with this idea is that if the malware signatures are too specific, then detection may miss malware that has a slight variation in its attack and if the specification is too broad then there is a risk of false positives [7].

Although this approach is commonly used by commercial anti-virus software companies, there has been concern about its effectiveness, the main concern being that it is vulnerable to polymorphism or obfuscation techniques because it depends on feature of the malware ([10], [11]). Another limitation of signature-based detection is that it is not suitable for identifying zero-day attacks, which is especially a problem given that malware can propagate quickly [12].

It is important to note, however, that although there are limitations with the existing detection approaches, there have

been efforts to overcome such limitations which include identifying characteristics which are common to the entire malware family as opposed to specific code sequences ([13],[14]).

Despite the limitations, a signature-based approach is not rejected in this study because each has its own method of detection and when used together they complement each other. For example, signature-based detection using a database will detect known worms and therefore, it would not be necessary to check all files and messages using behaviour-based detection, thus decreasing the workload on this part of the detection.

B. Behaviour-based detection

A response to the limitations of signature-based techniques has been behaviour based-techniques [15]. Behaviour-based detection uses the anomalous behavior of worms to detect worms by knowing what normal behavior is ([16], [12]). Unlike signature-based detection techniques, behavior-based techniques can describe the function and purpose of programs [17] and are capable of detecting future attacks partly because it is based on behavior [18] and therefore, can detect unknown worms unlike signature-based detection which can only detect previously known worms. Moreover, they do not require the use of a database because they only look for attempts by malware to replicate [19].

There are two main types of malware analysis within the behaviour-based approach, namely; static and dynamic analysis ([20],[21]) which are both used for detecting unknown malware [22], however, there is a limitation with dynamic analysis for detection because some worms are purely dynamic or purely static.

Specifically, when behaviour-based detection uses network analysis, an area that has been extensively researched [23], the detection is a two-stage process which involves the learning stage and the monitoring stage [24]. An example of this approach is NATE (Network Analysis of Traffic Events) used for detecting anomalous traffic [25].

The main limitation with behaviour-based techniques is that they depend on detecting a malware's attempt to replicate, if they miss this then damage will occur

C. Honey Pots

Honey pots are decoys which lure malware to carry out their behaviour away from valuable machines on a network[26]. The main purposes of honey pots is analysis and providing information about attacks [16], as well as an early warning system [26] and it is important to note that they do not replace internet security systems. Honey pots are used to collect information, such as how the malware replicates or propagates and a detailed analysis of their strategies; moreover, this information can be used to develop anti-virus software [22].

Because of the random nature of worms when they are looking for victims, honey pots are useful for detecting worms. More recently there has been the introduction of virtual honey pots used on virtual machines which have less risk; however, they are not suitable for real-world deployment [26]. Another limitation of honey pots is that they are often deployed on networks and will not collect malware that spreads through other means [27].

Because there are often a large number of malware attacks, honey pots often collect a large amount of data and it is essential that an appropriate algorithm is used to interpret the root cause of attacks [16].

III. Proposed Approach

The proposed approach of this study overcomes the limitations of existing worm detection systems. The novel contribution is to encourage the worms to cause damage in a dummy host and detect them based on their replication behaviour and the damage they cause. Using a dummy host has the additional benefit of protecting real hosts from exposure to a worm. A major limitation with other detection systems is that if they fail to detect the replication behavior of a worm then the worm will go on to cause damage to the host, however, with the WDS if it fails to detect replication behavior the resulting damage will not affect the real machines in the network and that damage is used as part of the detection process.

As mentioned in the above this study recognised the limitations of signature and behaviour-based detection mechanisms, however, also recognised the benefits, thus both were adopted in the proposed WDS. Therefore, these techniques will form part of the overall detection technique of the WDS. Moreover, this inclusion of signature-based detection means that the WDS will be comprised of a database of known signatures.

IV. Development of WDS

From the defining attributes of worms is that they propagate around networks, this was considered in the design of the WDS and where it would be located in the network. Subsequently, it was decided to integrate the WDS within the network as a hub and not with individual networked machines. This means that any traffic of files and messages between the networked machines would have to go through the WDS. For the purposes of experimentation the WDS was placed within a network of three machines (See Figure 1). The WDS is comprised of a number of components which include the networked machines, the Dummy Host, the Object Handler, the Database and the win32 folder. The design on the WDS was based on the use of ambients and before the WDS could be fully implemented using software it was first necessary to formalise the system.

A. Architecture

Because the architecture of the system required that it should be located within the network as a hub and all files and messages should pass through it, there were two main requirements, firstly, that it could manage the sending and receiving of files and messages between the networked hosts, and secondly, that it can detect worms through replication behaviour and damage. The components for these actions are described in the following:

B. Object Handler

The Object Handler is a middleware that has three main functions, firstly, to manage the sending and receiving of files

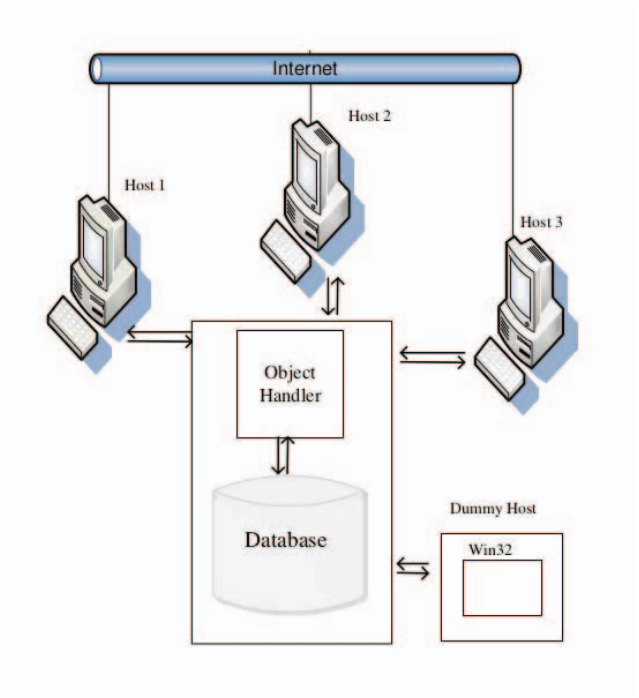


Fig. 1. Architecture of the Worm Detection System

and messages between the hosts via the WDS, secondly, to manage the communication that takes place related to detection, such as deny messages when files are found to be infected with a worm and thirdly, it is involved in the signature-based detection by sending and receiving files to the database to check for matching signatures. Because the Object Handler is located as a hub within the network it is not possible for any file or message to be sent from one machine to another without going through the Object Handler.

C. Database

The database contains a list of signatures of known malware and is used in the signature-based detection. The Object Handler will check a file or message with the database and if a signature is matched then it indicates the presence of a worm. When worms have been detected through the other detection mechanisms, i.e. using replication behaviour or damage as well as through signature-based detection, information about the newly detected worm is updated to the database. Specifically, the updated information contains the file identification, the name of the worm, worm signature and the location of the damage caused by the worm.

D. Dummy Host

The Dummy Host is responsible for detection of worms through replication behaviour and damage. The Dummy Host is one of the networked hosts (real machine) that acts as a safe environment which encourages worms to carry out their normal behaviour. This is achieved through allowing the worm to think that it is located in a normal machine. The WDS will detect a worm through replication and damage which is achieved through Hashing. There are two major benefits of using a Dummy Host, firstly, it encourages worms to carry out their behaviour, and secondly, it provides a safe and isolated environment for replication and damage to take place which protects the rest of the network.

V. Detection

A. Analysis

It was first necessary to carry out an analysis of worm behavior. This analysis was achieved using the PEiD, a software tool that unpacks the .exe, in order to derive information about the worm which includes whether or not the file is packed or unpacked and the name of its development platform; and the IDA pro disassembler which unpacks the binary code to reveal information about the worm's behavior.

The main behavioural trait that was derived from the analysis was replication. Replication is the characteristic that distinguishes worms from other types of malware and was therefore important for consideration in the development of the WDS.

Additionally from the analysis, a number of different worm behaviours were derived which included copy file where worms replicate themselves by copying an .exe file and creating a new file which caused damage by slowing down the system.

B. Classification

The results of the analysis lead to the development of a classification for worm behavior.This classification was based on the analysis of worms in the present study as well as other worm classification systems and classified worms according to replication and damage. Specifically, damage included the type and location of the damage and replication included the mode of replication. Both damage and replication were used in the WDS for detection purposes. Additionally, the researcher took ideas from classification systems by other researchers which included EDOWA which classified worms according to behaviour [28], specifically, methods of infection and propagation and Classification 2 [29] which considered methods of worm attack and defence and ways that the worm is detected.

VI. Detection Algorithm

The detection mechanism of the WDS is comprised of signature and behavior-based detection as well as detection using damage-based detection. The signature-base detection is achieved through a database, behavior-based detection is achieved by detecting the replication behavior of worms and damage detects worms through the damage that worms cause. The framework (algorithm) is presented in Figure 2

A. Detection using signature

The WDS comprises of an Object Handler that manages the sending and receiving of messages between networked hosts as well as managing the detection process. When a file is sent between two networked hosts the Object Handler will first check the signature of the file with the database against signatures of known malware, if a signature is matched then the presence of a worm is detected and the Object Handler will inform the sending host that the message is denied.

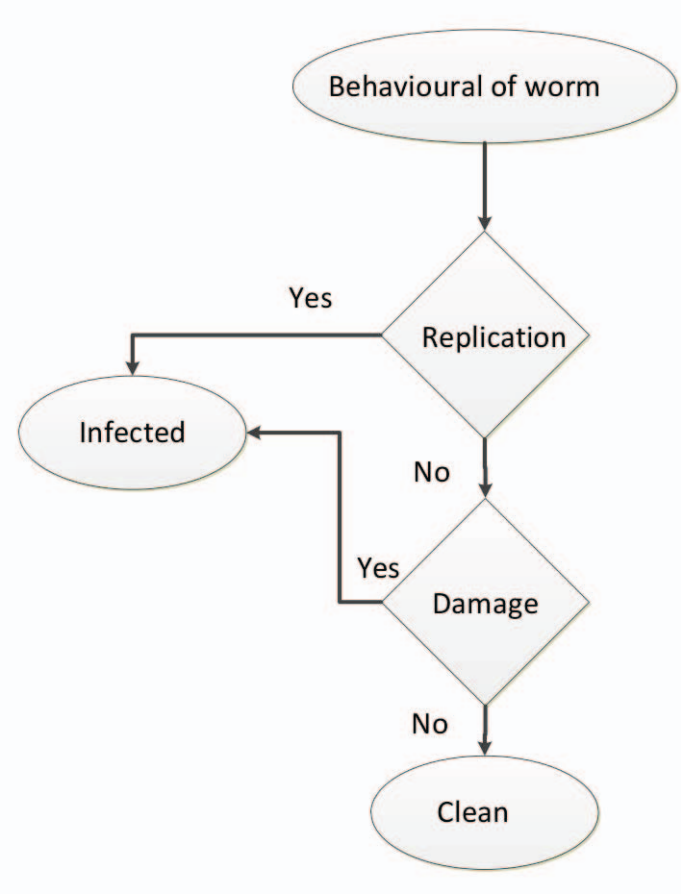


Fig. 2. Detection of Behaviour

B. Detection using behavior-based

If no worms are detected using signature-based detection, the Object Handler will send the file to the Dummy Host for behaviour-based detection. In the Dummy Host if there is a worm in the file then the worm will believe that it is in a real host and be allowed to conduct its normal behaviour, through which the worm will be detected. Specifically, the WDS detects the replication behaviour of worms which include replication in the same file, multiple self-replications in different locations and replication using different names. If the Dummy Host detects replication behaviour then it will inform the Object Handler and a deny message will be sent to the sending host. If no replication is detected the Dummy Host will then check for damage.

C. Detection using Damage

Detection of malware by detecting damage caused by malware is achieved through hashing in the Dummy Host. The Dummy Host sends the file to the Object Handler which then sends the file to the win32. In the win32 the file is checked for damage using hashing. Specifically, the win32 checks the condition newHash = myHash. If this condition is met then there is no change in the hash value and therefore, no worm is present, if there is a difference in the hash values this indicates the presence of a worm. Depending on the result either a CLEAN or INFECTED message will be sent to the Object Handler. The Object Handler then informs the sending host a message that the file or message cannot be sent due to the presence of a worm. The different types of damage that are detected include files being copied or created, change of file size and corrupted files, the WDS also reveals the location of the damage.

VII. Conclusion and Future work

In conclusion this study set out to overcome the limitations of current malware detection systems by adding a novel approach to detection. This novel approach, namely detection by damage, was successful in overcoming such limitations. Moreover, the use of a dummy host overcame the security weaknesses of other systems. It is a novel contribution that not only did it add a new approach to existing approaches but also expanded the use of dummy hosts for analysis of worm behavior to a detection mechanism.

Future work should include the development of a fully functioning WDS that can be integrated in a real-life network of machines. Moreover, the classification system of worms could be further developed to include more categories of replication behaviour and damage. Scalability is another area for future consideration, for example, the WDS could be developed to cope with larger networks with multiple simultaneous sending and receiving of files and messages. Furthermore, honey pots or dummy hosts are currently being used for worm analysis and authors of malware of finding new ways of recognising these hosts, therefore, future study could find techniques to hide the dummy host. Finally, future work could include applying the idea of this study to detecting other types of malware such as viruses.

References

[1] M. Dalla Preda and C. Di Giusto, "Hunting distributed malware with the κ-calculus," in *Fundamentals of Computation Theory*. Springer, 2011, pp. 102–113.

[2] D. Harley and R. Vibert, *AVIEN Malware Defense guide for the Enterprise*. Syngress Media Incorporated, 2007.

[3] D. Kleidermacher and M. Kleidermacher, *Embedded Systems Security: Practical Methods for Safe and Secure Software and Systems Development*. Newnes, 2012.

[4] J. Vacca and S. Ellis, *Firewalls: jumpstart for network and systems administrators*. Digital Press, 2005.

[5] C. Meadows, "Applications of formal methods to intrusion detection," *Encyclopedia of Cryptography and Security*, pp. 44–45, 2011.

[6] R. Shyamasundar, H. Shah, and N. N. Kumar, "Malware: from modelling to practical detection," in *Distributed Computing and Internet Technology*. Springer, 2010, pp. 21–39.

[7] J. Newsome and D. Song, "Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software," 2005.

[8] M. Ligh, S. Adair, B. Hartstein, and M. Richard, *Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code*. Wiley Publishing, 2010.

[9] B. M. Bowen, P. Prabhu, V. P. Kemerlis, S. Sidiroglou, A. D. Keromytis, and S. J. Stolfo, "Botswindler: Tamper resistant injection of believable decoys in vm-based hosts for crimeware detection," in *Recent Advances in Intrusion Detection*. Springer, 2010, pp. 118–137.

[10] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X.-y. Zhou, and X. Wang, "Effective and efficient malware detection at the end host." in *USENIX Security Symposium*, 2009, pp. 351–366.

[11] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario, "Automated classification and analysis of internet malware," in *Recent Advances in Intrusion Detection*. Springer, 2007, pp. 178–197.

[12] W. Fan, M. Miller, S. Stolfo, W. Lee, and P. Chan, "Using artificial anomalies to detect unknown and known network intrusions," *Knowledge and Information Systems*, vol. 6, no. 5, pp. 507–527, 2004.

[13] M. Christodorescu and S. Jha, "Static analysis of executables to detect malicious patterns," DTIC Document, Tech. Rep., 2006.

[14] M. Christodorescu, S. Jha, S. Seshia, D. Song, and R. Bryant, "Semantics-aware malware detection," in *Security and Privacy, 2005 IEEE Symposium on*. IEEE, 2005, pp. 32–46.

[15] R. Sommer, D. Balzarotti, and G. Maier, *Recent Advances in Intrusion Detection: 14th International Symposium, RAID 2011, Menlo Park, CA, USA, September 20-21, 2011, Proceedings*. Springer, 2011, vol. 6961.

[16] F. Pouget, M. Dacier *et al.*, "Honeypot-based forensics," in *AusCERT Asia Pacific Information Technology Security Conference*, 2004.

[17] M. Hypponen, "Malware goes mobile," *Scientific American*, vol. 295, no. 5, pp. 70–77, 2006.

[18] K. Ramachandran, *Stochastic and epidemiological models for performance evaluation of peer-to-peer networks*. ProQuest, 2007.

[19] J. Morales, P. Clarke, and Y. Deng, "Identification of file infecting viruses through detection of self-reference replication," *Journal in computer virology*, vol. 6, no. 2, pp. 161–180, 2010.

[20] B. B. H. Kang and A. Srivastava, "Dynamic malware analysis," in *Encyclopedia of Cryptography and Security*. Springer, 2011, pp. 367–368.

[21] P. Beaucamps, I. Gnaedig, and J.-Y. Marion, "Behavior abstraction in malware analysis," in *Runtime Verification*. Springer, 2010, pp. 168–182.

[22] Q. Zhang, "Polymorphic and metamorphic malware detection," Ph.D. dissertation, North Carolina State University, 2008.

[23] T.-F. Yen and M. K. Reiter, "Traffic aggregation for malware detection," in *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2008, pp. 207–227.

[24] A. Moser, C. Kruegel, and E. Kirda, "Limits of static analysis for malware detection," in *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*. IEEE, 2007, pp. 421–430.

[25] C. Taylor and J. Alves-Foss, "Nate: N etwork analysis of a nomalous t raffic e vents, a low-cost approach," in *Proceedings of the 2001 workshop on New security paradigms*. ACM, 2001, pp. 89–96.

[26] N. Provos, "Honeyd-a virtual honeypot daemon," in *10th DFN-CERT Workshop, Hamburg, Germany*, vol. 2, 2003.

[27] S. Poeplau and J. Gassen, "A honeypot for arbitrary malware on usb storage devices," in *Risk and Security of Internet and Systems (CRiSIS), 2012 7th International Conference on*. IEEE, 2012, pp. 1–8.

[28] M. M. Saudi, E. M. Tamil, S. A. M. Nor, M. Y. I. Idris, and K. Seman, "Edowa worm classification," in *Proceedings of the World Congress on Engineering*, vol. 1, 2008.

[29] A. Pratama and F. A. Rafrastara, "Computer worm classification," *International Journal of Computer Science*, vol. 10, 2012.