IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 1, Ver. I (Jan.-Feb. 2017), PP 31-49
www.iosrjournals.org

Computer Worms Based on Monitoring Replication and Damage: Experiment and Evaluation

Yazed B. Al-Saawy (1), Sulaiman Al amro (2)
[1](Information Technology (IT) Department, Islamic University in Madinah, KSA)
[2] (Computer Science (CS) Department, Qassim University, Qassim, KSA)

Abstract: *This paper presents the experiments of the proposed worm detection system WDS and its evaluation. More specifically, initially there will be an explanation of the various experiment designs and how the experiments will be conducted. The results are presented and an evaluation will take place against a set of predetermined criteria. The experiments involve networking three machines over wireless links and transferring files between them which may contain worms in order to test the W DS. The three machines are Host 1, Host 2 (Dummy Host) and Host 3. The evaluation of the system showed that all evaluation criteria were successfully met.*

Keywords: *Computer worms, Damage, Replication, Malware detection, Worm detection system.*

I. Introduction

Organisations are becoming increasingly networked and their entire purpose and functionality depend on such networks. However, with an increase in networked systems there is an associated increase in the risk of malware, specifically worms, moving across these systems and causing damage. Worms are malware that proactively propagate across networks and exploit vulnerabilities in programs and operating systems [24]. Because so many industrial and business operations depend on these networks, the damage can be costly and widespread and can even threaten the existence of organisations. There are efforts to protect organisations from such malware attacks, one of which is malware detection. Currently, malware detection systems depend on mechanisms which include detection by malware behaviour and signature-based detection [26, 25]. Unfortunately, the latter detection method, signature-based, is not 100 per- cent efficient because it depends on gathered information about the signatures of previously detected malware and are not capable of detecting new unknown malware. A response to the problems of signature-based detection has been behaviour-based malware detection; the approach depends on monitoring a system for changes in behaviour which may indicate the presence of malware. Unfortunately, the main problem with this approach is that infection has already taken place which may already be too late. This paper is an extension for the paper presented by Al-Saawy and Al amro (2016) [1].

II. Literature Review

There is a review of what worms are their anatomy and their mode of operation which includes how they propagate and the damage that they cause to machines. Moreover, there is a review of the literature pertaining to existing worm classification systems. Importantly, existing detection mechanisms including their benefits and limitations is also reviewed. Finally, part of the development of the proposed WDS involves formalisation, and therefore, there is a review of available modelling formalisms.

II. I Detection Approaches

Because the present study is concerned with the development of a novel approach to detecting worms, it is necessary to review the literature related to existing worm detection approaches. Specifically, a review of the literature will serve a number of purposes. Firstly, the review will serve to highlight the limitations of current detection approaches to further justify the need for the development of a new approach to detection, secondly, to reveal issues that would be relevant to the development of the detection approach and finally, will contribute to the development of criteria for the detection approach of this study, in addition to other primary and secondary research.The threat of malware attacks is an unavoidable problem in computer security [2]. Threats have grown exponentially and the currently used signature and anomaly-based (behaviour-based) techniques are turning out to be obsolete [3]. As new vulnerabilities and attack methods are discovered, anti-malware technology quickly becomes obsolete because fundamentally, the anti-malware software is a re- active technology [4] and thus, there is a need for a new approach to overcome these problems.To counter these threats, malware analysis has become a vital undertaking to- wards developing effective detection and removal techniques. However, successful analysis of a system should not only be limited to detecting an attack for it should adequately provide information about how the attack takes place and the consequences of the infection

[5]. This idea is echoed by Zelster [6] who said that detection may need to be conducted as part of both the incident response and the forensic analysis processes. In relation to this idea, one of the contributions of the present study is that it extends the analysis process to detection by using a Dummy Host not for analysis but for detection. Investigating malware behaviours and mechanisms are useful in designing and implementing preventive mechanisms both on computer and data networks [7].

The techniques for detecting malware can be classified into two broad categories: anomaly-based (behaviour-based) and signature-based. Anomaly-based detection refers to using pre-existing knowledge of what construes 'normal' behaviour for the computer system to decipher if a malicious entity is in operation or not [8]. A subset of the anomaly-based detection technique is the specification-based detection technique. These reference a predefined rule-set or 'specification' of what is accepted as valid behaviour from the systems components, and any program that violates these specifications is considered to be malware. On the other hand, signature- based detection techniques use pre-defined characterisations of what is defined to be malicious to decide the validity of the process/program under inspection.Figure 1 shows the three different techniques of malware detection. As can be seen, each of the three techniques can follow one of three different approaches, e.g., static, dynamic or hybrid which are now discussed in detail below.

As noted above, each technique can use a static, dynamic or hybrid approach to detection. According to Idika and Mathur [8] "The specific approach or analysis of an anomaly-based or signature-based technique is determined by how the technique gathers information to detect malware. Static analysis uses syntax or structural properties of the program (static)/process (dynamic) under inspection (PUI) to determine its maliciousness". Idika and Mathur ([8], Pg. 5) further say in relation to this that a static approach to signature-based detection leverages structural in- formation to uncover the maliciousness, whereas a dynamic approach will leverage runtime information. Generally speaking, the static approach will attempt to detect malware before it can execute, while the dynamic approach promotes detection during or after the execution of the malicious code. The hybrid approach, as the name suggests, combines elements of both approaches [9].

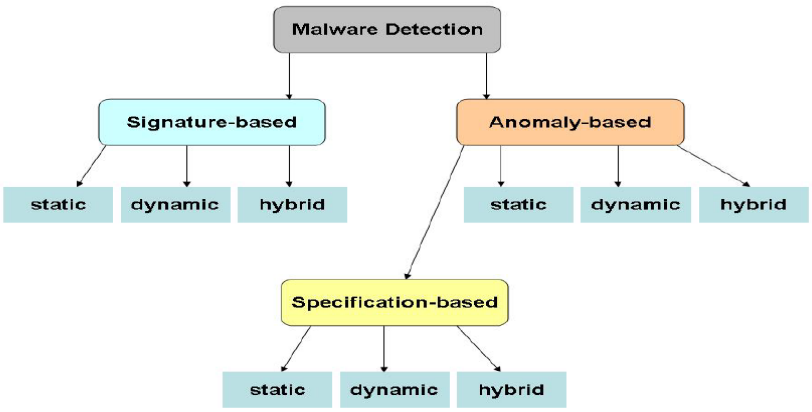


Fig.I: Classification of Malware Detection Techniques [8]

• Static Analysis

The proposed detection system in the present study comprises of two layers of detection, signature-based and behaviour-based, and uses static analysis for both. Signature-based detection is a detection approach and uses static analysis for pattern matching. As for the behaviour-based layer of detection, static analysis is preferred by researchers over dynamic analysis.Static analysis offers a significant improvement in malware detection accuracy compared to the traditional pattern-matching [10]. Typically syntactic in nature, traditional malware detectors (also known as signature-based detectors) use pattern matching to compare the byte sequence comprising the body of the malware against a signature database [11]. Theoretically, static analysis can cover the complete program code, examine all possible paths of execution and is usually faster than dynamic analysis [12]. Furthermore, in the context of formal analysis, researchers derive certain advantages from static analysis and thus favour it over dynamic analysis. Beaucamps [13] has provided two major advantages which may be relevant to a detection approach that proposes to use behaviour analysis: (1) behavioural analysis of a program can be done in a more exhaustive way for it analyses the unbounded set of the program execution traces or an approximation of it; and (2) static behaviour analysis complements classical, dynamic and behaviour analyses of some future behaviour at some critical point in an execution. In contrast, dynamic behaviour analysis only analyses the past behaviour, and because the captured data in such a case may not be sufficient to raise an alarm, such oversight may let the program execute the critical operation [13]. This would be a disadvantage in a detection system.

Moreover, according to Kang and Srivastava [14] this method requires analysis of the malware code which is usually not available for analysis; however, even if it is available, the malware often obfuscates the code, making the static analysis difficult and time-consuming. And while static analysis possesses the potential to reason about possible problem behaviour of software, the underlying computation problems are difficult, especially when working with binary code [15].Beaucamps [16] also bring attention to a limitation of static analysis which is the general intractability of knowing in advance the entire program code as it may change dynamically, this idea is supported by Rieck [10] who say its main weakness lies in the difficulty to handle obfuscated and self-modifying code, and Zhang [12] who says it is especially difficult when attackers deliberately craft malware that are difficult to analyse statically [12]. Moreover, Preda [11] says that in determining how code mutates, traditional static analysis techniques are inadequate as they assume that programs do not change during their execution further contend that metamorphic viruses can reprogram and develop sophisticated code obfuscation techniques, making it more difficult for static analysis techniques and can also overcome dynamic analysers by changing its behaviour.

However, there has been a response to the aforementioned problems associated with static analysis. Founded on formalisms, Christodorescu and Jha [17] developed a prototype called SAF E (Static Analyzer for Executables) designed to detect malicious patterns in executables that are usually resistant to normal obfuscation transformation. Shyamasundar [18] acknowledged that their work yielded interesting approaches to detecting variations of a known virus by carrying out static analysis on its code and deriving an abstraction of its behaviour.

• Dynamic Analysis

In the above it was shown that a significant problem of approaches for static analysis was that malware could obfuscate the code, however, dynamic analysis is said to be immune from this [14], [19] and offers a promising way to analyse malicious software [20]. As a result, most practical malware analysis techniques have focused on dynamic analysis [15] that analyses the code of a program by actually executing it [12]. As such, the actual execution of a malware sample is an important part of the analysis process [20] and is concentrated on gaining accurate and reliable information when malicious programs are executed [10]. According to Kang and Srivastava [14] dynamic analysis can be done by firstly, taking the system state image before malware execution and then comparing it with the system state after the malware execution which provides details about the mal-ware at an abstract level, and secondly, executing the malware and observing its behaviour during execution. This approach provides behaviour details at a much more granular level as it involves the monitoring of malware's interaction with the registry, the file system, as well as the network.

However, dynamic analysis has limitations too. Gorecki [20] say it exposes the runtime environment to malicious activity and may not be feasible in many cases since malware often needs Internet access to trigger and exhibit a malicious behaviour that is to be analysed. Dunham [21] says it is restricted to seeing only one run of an analysed sample and Zhang [12] says it covers only a part of all possible program execution paths. Moreover, Zolkipli and Jantan [7] bring attention to the limited coverage of analysis unless multiple paths are triggered, and Grobert [39] says that the approach has the general constraint that if code is not executed, it cannot be analysed; and finally, Zhang [12] says that dynamic analysis incurs more overheads than static analysis; difficulty in simulating the execution conditions un- der which the analysed malware exhibits its malicious behaviours; and, if the code is executed in a virtual machine, there are techniques that attackers can utilise to redesign the code and modify its runtime behaviour.

• Hybrid Analysis

Literature on de-obfuscating techniques [22]; automating the hidden-code ex- traction of unpack-executing malware [24] and designing type-checking systems for integrity of the data flow in Windows Vista [23] have presented a way to combine static and dynamic analysis with the purpose of acquiring the advantages found on both procedures and develop a more explicit analysis, more powerful evaluations and/or obtain greater levels of soundness. Chaudhuri [23] contend that the method of combining dynamic access control with static typing for protection is similar to hybrid type-checking and is a powerful technique. Royal [24] assert that such an approach produces a detection algorithm that overcomes the deficiencies of other approaches. Lastly, Preda [22] has cited that combining both program analyses provides a set of heuristics for overcoming important obfuscating techniques.

III. Experiment Design

The experiments aim to demonstrate that the system can detect worms using the unique approach of detecting replication behaviour or damage. The design of the experiments considers what these experiments are designed to achieve. Specifically, the experiments were designed to establish whether or not the system was able to detect different types of worms, including unknown and known worms through detecting predetermined types of damage and replication. How this is achieved through the experimentation will be explained providing the details of each of the experiments and the associated detection technique. The system is designed to detect a

specified number of damages; worms that cause damage which is not specified will not be detected by damage but may be detected by replication.

III. I Detection Approaches

In these experiments we are testing whether the proposed WDS can detect known worms through damage or replication. There are five known worms in total, each causing a specific type of damage. The experiment will allow the worm to move through the system. This will show that the network of the system and the communication processes between various system components are working properly while the infection and detection of the worm are taking place. The experiment will also show that the known worms are able to replicate and inflict damage in the Dummy Host and whether the system is able to detect the worms through that replication and damage. Each known worm will be introduced to the system separately. Al- though the experiments for running and testing the known worms are the same, the detection of each of the known worms involves detecting different types of damage as indicated in Table 1. The different types of damage and replication behaviour were derived from the classification, and the experiments here are designed to test if the WDS is able to detect these types of damage and replication behaviour.

Table I: Types of Damage

	Type of Damage
Type 1	Corrupts file
Type 2	Destructive file.
Type 3	Create files.
Type 4	Change files size.
Type 5	Copy files

The experiment will establish if replication behaviour takes place and uses that behaviour in the detection process, specifically, there are three types of replication behaviour that are determined by the WDS as presented in Table 2

Table II: Types of Replication

	Types of Replication
Replication Type 1	Replication in same location with different name.
Replication Type 2	Replication in different location with same name.
Replication Type 3	Replication in different location with different name.

III. IIDetecting Unknown Worms

The experiment will also determine whether or not the system is capable of detecting unknown worms. Detection of unknown worms is achieved by one or a combination of two methods. The first method is detecting damage types which are already known to the system and the second method of detection is by replication behaviour detecting a change in the Hash Directory of the folder (Win32) which contains the system files.

Like with the experiments for known worms, this experiment will involve evaluating various mechanisms of the system which includes communication between the components such as the Agents, the Object Handler and the Dummy Host. This evaluation will take place while the system is engaged in detecting worms.

III. III Detecting Known Worms through Signatures

The signature-based detection offers an additional layer of detection to the overall detection mechanism and acts to filter files with existing known signatures. By filtering infected files in this way, it is not necessary to send these files to the Dummy Host for detection which decreases the Dummy Host's workload. The system should be able to detect known worms from their signatures by checking against signatures of previously known worms compiled in the database. Specifically, the experiment will test if a known worm with a known signature will be matched in the database, and therefore, this will show whether or not this particular part of the detection mechanism works. As with the other experiments, this experiment will also test the functioning of the system's mechanisms, this is particularly important here because it requires that the database is correctly positioned in the system and appropriate communication is taking place between the database and other system components.

IV. Evaluation Criteria

In any evaluation of a system a number of evaluation criteria need to be established, against which the system will be evaluated. There are a number of different functions that need to work in order for the overall WDS to be successful. The evaluation criteria are established based on the overall requirements of the functionality of the system and are as follows:

- The unique feature of the WDS is that it allows replication behaviour and damage to take place and then detects a worm based on the detection of that replication behaviour or damage. Therefore, one of the criteria is that the WDS allows a worm to infect files in the Dummy Host and that subsequent replication or damage

is allowed to take place. It is important to note that for known worms the types of replication and damage are prespecified.

- The system depends on detecting the replication or damage that has occurred in order to detect the presence of a worm. Therefore, the criterion here is that the system is able to detect replication or damage.
- It is a requirement of the system that it can detect worms that are already known and have known signatures. Therefore, it is a criterion that the WDS successfully detects worms by matching signatures in the database that contains a directory of known signatures.
- The system should be capable of detecting unknown worms; therefore, one of the criteria is that the system can detect unknown worms through detecting prespecified damage, or failing this, through replication behaviour.

V. Running Experiments

Experiments are conducted for each of the known and unknown worms. Each experiment involved checking the results of the console interfaces to see if the system processes which include communication, detection and updating the database, function correctly. In total, there are four consoles which represent Hosts 1, 2 and 3 and the database. Host 1 being the sending agent, Host 2 the Dummy Host (DH) and Host 3 the receiving agent and an additional console for the database. For experiments where detection does not take place and the file is clean all four consoles will be shown, this is because there will be an activity in each of the three hosts in this case. For experiments where detection of a worm does occur, the consoles for Host1 and DH as well as the console for the database will be shown since the file will not reach Host 3 and there will be no activity to show. Moreover, in experiment 1 the GUI will be shown to demonstrate that communication processes including the IP addresses and the ports are functioning correctly.

The experiment involves running the system by sending files and messages be- tween the hosts, and depending on which specific experiment is taking place, these files will contain known worms, unknown worms or no worms. Moreover, as mentioned above, the experiments are designed to determine whether files and messages can successfully be sent and received, whether there is a successful detection by various means, e.g., damage, signature and replication and subsequent updating in the database.Additionally, the experiment involves determining if the various processes of the system work. For example, when a file containing a worm is sent from Host 1 to Host 3 via the Object Handler, the experiment will check if the specific processes took place, e.g., that the worm was detected, and that a deny message was sent back to the sending host. The following section presents the experiments.

V. I Experiment I

The purpose of Experiment 1 is to firstly determine if the system functions are work- ing properly and that the consoles are displaying the correct information including reply and deny messages and file destination. The main purpose of Experiment 1 is to send a file from Host 1 to DH and to determine if successful detection takes place. It is also an aim of the experiment to demonstrate that the system allows replication behaviour and damage to occur in the Dummy Host (DH) and is able to detect a worm through that replication behaviour or damage. The experiment also aims to check that updating the database with information about the detected worm is successful.The expected outcomes are that the consoles clearly illustrate the sending and receiving of a file and subsequent information upon detection such as deny messages. Moreover, it is expected that the detection system will detect a worm successfully through detecting replication behaviour or damage.In Experiment 1 the GUI interface is executed and shows that the functions for the entire system are working correctly. This GUI was placed in Host 1 (see Figure 2).

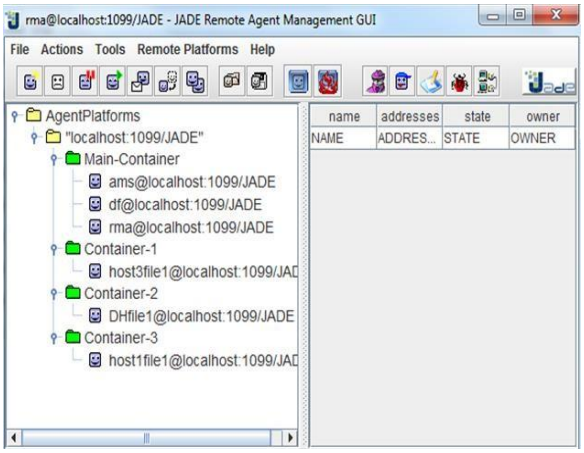


Figure II: Experiment GUI Interface Results for Host 1 Console 1

For each experiment the interface consoles are used to display the results of the processes. The console for Host 1 in list 1 shows that Host 1 successfully displays relevant information relating to File1 which included the file destination and the various information related to the sending of the File. Moreover, the name of the file being sent was displayed as DH File1 in the DH Console (list 2. The agent ID of the receiving agent (DHfile1@localhost:1099/JADE) was also identified. Following this, the console showed that the message had been sent by Host 1. In this experiment a worm was detected through replication behaviour and damage which is also indicated in the Host 1 console and the corresponding message 'file got infected' is also returned by DHFile 1. Specifically, the replication behaviour was Type 3 and the cmd.exe was damaged.

Listing I: Host 1 Console

```
1 Hello. My name is host1file1
2 the File Name c:\\yaz\\cmd.exe
3 the File change 0
4 the message had been sent
5 this is called from the DptW32blaster2
6 == Answer<- cmd.exe file got affected from DHfile1@localhost:1099/JADE
```

Results for DH Console 1

The following console shows that the file was successfully sent from H ost1 to DH .

Listing II: Host 1 Console

```
1 Hello. My name is DHfile1
2 ********** This is before the replication Start*********
3 Listing files found for Hash Directory
4 -- c:\yaz\ScrapInsertedObj.exe
5 -- c:\yaz\cdrom.sys
6 -- c:\yaz\cmd.exe
7 -- c:\yaz\originalScrapInsertedObj.exe
8 -- c:\yaz\originalcdrom.sys
9 -- c:\yaz\originalcmd.exe
10 -- c:\yaz\originalwinload.exe
11 -- c:\yaz\winload.exe
12 Hash value for the directory: 0562fd4f526d372cb59997fdnhjui87
13 The value of the Replication Type 3
14 ********** This is After the replication Start*********
15 Listing files found for Hash Directory
16 -- c:\yaz\ScrapInsertedObj.exe
17 -- c:\yaz\cdrom.sys
18 -- c:\yaz\cmd.exe
19 -- c:\yaz\originalScrapInsertedObj.exe
20 -- c:\yaz\originalcdrom.sys
21 -- c:\yaz\originalcmd.exe
22 -- c:\yaz\originalwinload.exe
23 -- c:\yaz\winload.exe
24 Hash value for the directory: 0562fd4f526d372cb59997fd83742f56
25 Database Connection successfully made
26 the SqlQuery Insert into FilesInfo (id, name, signature, locationOfTypeDamage) values (0,'File1
','401522675d495380d8bc73cae9dd1d01','In cmd.exe is corrupt');
cmd.exe is corrupt');
27 EE8CBF12D87C4D388F09B4F69BED2E91682920B5
28 8804021 A17895FF6D78CC31E47640336064C8900
29 File Has changed
30 Original File Hash value EE8CBF12D87C4D388F09B4F69BED2E91682920B5
31 Current file Hex value 8804021 A17895FF6D78CC31E47640336064C8900
32 C:\ yaz0\cmd0.exe Written
33 C:\ yaz1\cmd1.exe Written
34 C:\ yaz2\cmd2.exe Written
35 The cmd.exe file replicated 3 times
36 the signature of the file is 401522675 d495380d8bc73cae9dd1d01
37 the Reply to the Sender ( agent -identifier :name host1file1@localhost :1099/ JADE :addresses (sequ
38 Message had been sent
```

The initial part of the detection process checks for matching signatures in the database and no signatures are matched. Subsequently, the Object Handler success- fully sent the file to the Dummy Host.
A change in the Hash directory for the folder indicates there is replication behaviour in a file contained in the folder. Replication by a worm took place in the Dummy Host, specifically, this replication created multiple files of cmd.exe with different name extensions and different locations such as C: yaz0 directory, namely;
'cmd0.exe' and 'cmd1.exe'. This indicated that the 'cmd.exe' was replicated and damaged and subsequently a worm, worm 1 was detected (see list 2) by the Dummy Host. Dummy Host (DH) successfully sent a deny message to Host 1.

- **Results for Updating the Database**

The experiment involved checking if updating the database was successful. The DH console shows that the detected worm was successfully updated in the Database. This included the signature and file id of the worm in the signature table of the database and the name of the worm, the worm's signature, the type of damage, damage location and the file ID in the file info table of the database. The following is the console log for updating the database (see Figure. 3).

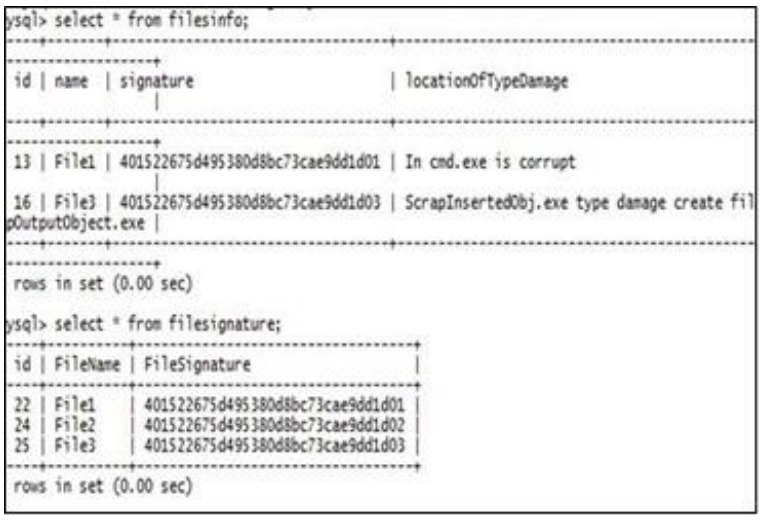


Figure III: Updating the Database Console

- **Results for Host 3 Console 1**

The console for Host 3 is shown in Figure 4 and it illustrates that it does not receive any files or messages and its status was shown to be inactive.

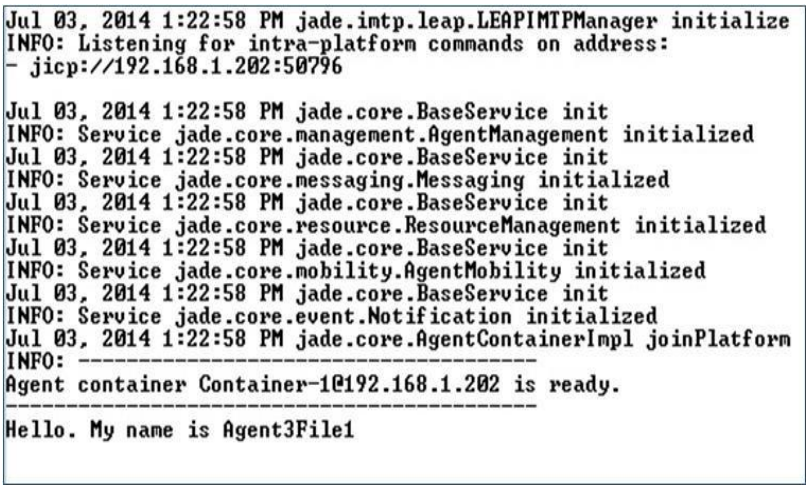


Figure IV: Host 3 Console 1

- **Verification of Results**

To provide further verification that replication and damage really took place; the following five parameters are displayed by the system in the consoles. These parameters are provided for each of the subsequent experiments and are shown below in Table 3.

Table III: Verification of Results 1

Name	Location of replica ion and damage	Type of Replication	Damage type	File status
W32blaster (Worm1)	cmd.exe	3	Corrupts cmd.exe	Infected

Additionally the researcher also uses time stamp and the 'cmd properties' to verify the damage; for example, when 'cmd.exe' is damaged the icon changed and when the properties for the file were checked the time stamp revealed the time of the damage (see Figure 5).

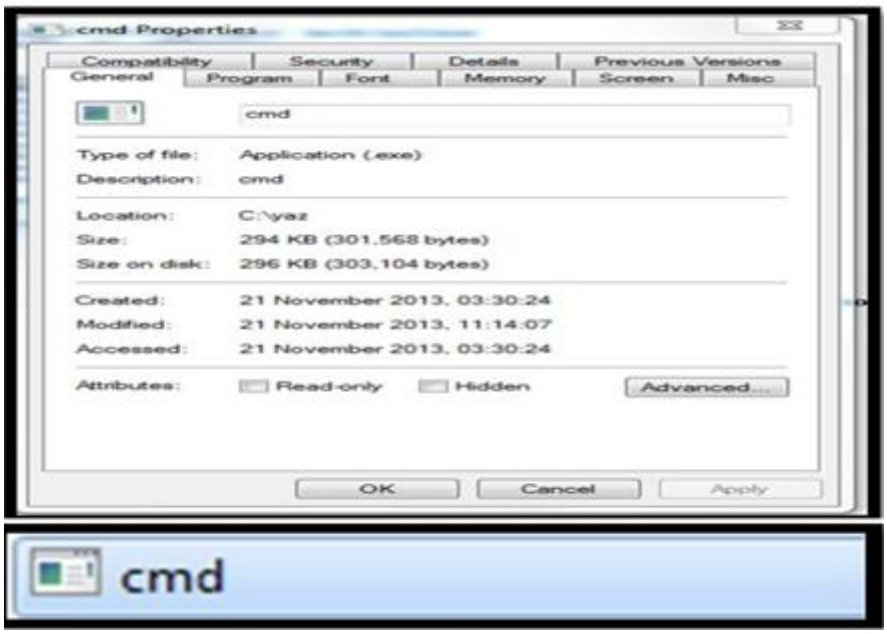


Figure V: DH Console 1 Verification

V. II Experiment II

There are a number of reasons for carrying out Experiment 2, first to check if the system components are communicating with each other properly and the consoles are displaying the correct information including file destination and messages that indicate either good or bad files. The main part of this experiment is to send a file from Host 1 to Host 3 via the DH and to verify if the detection mechanism is working. Like Experiment 1, Experiment 2 also aims to determine if the Dummy Host allows replication behaviour or damage to take place in the event that there is a worm present. Here the replication behaviour was type 2 and the damage was to corrupt the file. The experiment should demonstrate that the detection functions are working correctly. Finally, the experiment will check to see if the update func- tion in the database is also working.

- **Results for Host 1 Console 2**

The Host 1 console successfully displays the name of the file (cmd.exe see List 3) being sent, confirming that the file was sent to DH and the reply from DH which indicated the status of the file, in this case 'cmd.exe file had been created from DHfile2' (see List 3) was sent to Host 1 from DH.

Listing II: Host 1 Console 2

```
1 the File Name c:\\ yaz \\ CMD.exe
2 the File change 0
3 Hello. My name is host1file2
4 the conversation ID DptTrapDoors1
5 the value of the dataToParse c:\\ yaz\\ CMD.exe|0
6 the message had been sent
7 this is called from the Dh
8 == Answer <- cmd.exe file had been created from DHfile2@146 .227.115.149:1099/ JADE
```

- **Results for DH Console 2**

Experiment 2 aims to test if replication behaviour and damage is allowed to occur in the Dummy Host in the case a worm is present and if that replication or damage is detected.

Listing IV: DH Console 2

1 Hello. My name is DHfile2
2 ********** This is before the replication Start **********
3 Found files for hashing:
4 – c:\yaz\ ScrapInsertedObj .exe
5 – c:\yaz\Test.txt
6 – c:\yaz\cdrom.sys
7 – c:\yaz\cmd.exe
8 – c:\yaz\ originalcdrom .sys
9 – c:\yaz\ originalcmd .exe
10 – c:\yaz\ originalwinload .exe
11 – c:\yaz\winload.exe
12 Hash value for the directory: 4727 ce9976fbedc5c394a9ba99269be1
13 the Destructive File.exe
14 ********** This is After the replication Start **********
15 Found files for hashing:
16 – c:\yaz\ ScrapInsertedObj .exe
17 – c:\yaz\ Destructive File.exe
18 – c:\yaz\Test.txt
19 – c:\yaz\cdrom.sys
20 – c:\yaz\cmd.exe
21 – c:\yaz\ originalcdrom .sys
22 – c:\yaz\ originalcmd .exe
23 – c:\yaz\ originalwinload .exe
24 – c:\yaz\winload.exe
25 Hash value for the directory: 4727 ce9976fbedc5c394a9ba99456yhh
26 Database Connection successfully made
27 the SqlQuery Insert into FilesInfo (id , name , signature , locationOfTypeDamage) values (0,'File2 ',
'this2Destructive File.exe ','This is about Destructive File.e
28 xe ');
29 C:\ yaz0\ Destructive File.exeWritten the Destructive File.exe file replicated 1 times in the diffe
30 the Destructive File.exe file replicated 1 times
31 the signature of the file is this2Destructive File.exe
32 the file is affected true

The console for DH shows that the file is successfully received by the DH. Initial detection is carried out by matching known signatures in the database and no sig- natures were matched. Subsequently, the Object Handler sent the file to the DH.

The console for DH also shows the list of files in the C://yaz folder and the Hash directory before infection takes place, the console also shows a list of the same files in the C://yaz folder after the file has been received (see List 4). The Hashing showed that there was a change in the Hash directory which indicated that there was an issue with a file contained within the folder. Specifically, the DH console shows in the second list, after the file was received, that a new file named 'Destruc- tive File' had been created which was subsequently duplicated. The first list showed (C://yaz//Destructive File.exe) and the second list showed (C://yaz0//Destructive File.exe), indicating the creation of a new file and replication and thus the presence of a worm.

The DH console indicates that a reply message was sent to Host 1. The Host 1 console shows the message was received, indicating that a file had been created (see List 3). The console for DH also shows that the database was successfully updated with the name of the worm, the worm's signature, type and location of damage and the file ID.

Table IV: Verification of Results 2

Name	Location of replication and damage	Type of Replication	Damage type	File Statu
TrapDoors (Worm2)	cmd.exe (new file replicated)	2	Created New file Destructive File.exe	Bad

V. II Experiment III

The main aim of Experiment 3 is to test if the system was able to detect a worm through replication behaviour and damage. More specifically, the experiment will test if the worm is allowed to carry out replication and damage in the Dummy Host. Additionally, the experiment will test if the system functions are working as expected and that the consoles are displaying the correct information. The experiment will involve sending a file

from Host 1 to Host 3 via the DH to see if infection takes place. The experiment should demonstrate that the detection functions and the update function (in the database) are working correctly.

- **Results for Host 1 Console 3**

Host 1 console for Experiment 3 successfully displays the name of the file that was sent, confirmation that the file was sent to DH and the reply from DH showing the status of the file. Here the reply message was 'ScrapOutputOb ject.exe file had been created' indicating that the detection in DH detected a worm, (see List 5).

Listing V: Host 1 Console 3

1 Hello. My name is host1file3
2 the File Name c:\\ yaz \\ ScrapInsertedObj .exe
3 the File change 0
4 the Agent id (agent -identifier :name DHfile3@146 .227.115.149:1099/ JADE)
5 the conversation ID DptWormProlaco1
6 the message had been sent
7 this is called from the DptWormProlaco2
8 == Answer <- ScrapOutputObject .exe file had been created from DHfile3@146 .227.115.149:1099/ JADE

- **Results for DH Console 3**

The experiment is run with File 3 (ScrapInsertObject.exe) and the DH console shows that File 3 was successfully received by the DH. The console also shows that replication and damage occurred in the DH and that a worm is successfully detected by that replication behaviour and damage. The experiment shows that detection was achieved through a change in the Hash directory which indicated the presence of a worm. More specifically, the console shows that the worm was able to infect DH by disguising itself as an exe file named (C://yaz//ScrapInsertObject.exe), the system was able to detect this worm by detecting the output operation shown as: (C:/yaz//ScrapOutputObject.exe) which was subsequently replicated with different names, e.g., C:/yaz//ScrapOutputObject0.exe and C:/yaz//ScrapOutputObject1.exe etc. (see List 6). Thus the type of replication was Type 1 and the damage was to create file.

Listing VI: DH Console 3

1 Hello. My name is DHfile3
2 ********** This is before the replication Start *********
3 Found files for hashing:
4 -- c:\yaz\ ScrapInsertedObj .exe
5 -- c:\yaz\ ScrapOutputObject .exe
6 -- c:\yaz\TaskMon.exe
7 -- c:\yaz\Test.txt
8 -- c:\yaz\cdrom.sys
9 -- c:\yaz\cmd.exe
10 -- c:\yaz\ originalcdrom .sys
11 -- c:\yaz\ originalcmd .exe
12 -- c:\yaz\ originalwinload .exe
13 -- c:\yaz\winload.exe
14 Hash value for the directory: a7d31e598a97415bb61334ca38910b94
15 the ScrapOutputObject .exe
16 ********** This is After the replication Start *********
17 Found files for hashing:
18 -- c:\yaz\ DetectionHandlerOutput .txt
19 -- c:\yaz\ ScrapInsertedObj .exe
20 -- c:\yaz\ ScrapOutputObject .exe
21 -- c:\yaz\ ScrapOutputObject0 .exe
22 -- c:\yaz\ ScrapOutputObject1 .exe
23 -- c:\yaz\ ScrapOutputObject2 .exe
24 -- c:\yaz\TaskMon.exe
25 -- c:\yaz\Test.txt
26 -- c:\yaz\cdrom.sys
27 -- c:\yaz\cmd.exe
28 -- c:\yaz\ originalcdrom .sys
29 -- c:\yaz\ originalcmd .exe
30 -- c:\yaz\ originalwinload .exe
31 -- c:\yaz\winload.exe
32 Hash value for the directory: 52 b5b8423ec502bbb5a5141e1c010015

33 Database Connection successfully made
34 the SqlQuery Insert into FilesInfo (id , name , signature , locationOfTypeDamage) values (0,'File3 ','this2ScrapOutputObject .exe ','This is about ScrapOutputObject
35 .exe ');
36 C:\yaz\ ScrapOutputObject0 .exe Written
37 C:\yaz\ ScrapOutputObject1 .exe Written
38 C:\yaz\ ScrapOutputObject2 .exe Written
39 the signature of the file is this2ScrapOutputObject .exe
40 the file is affected true
41 the Reply to the Sender (agent -identifier :name host1file3@146 .227.115.149:1099/ JADE: addresses (sequence http ://146.227.115.149:7778/ acc))

The DH console shows that a reply message is successfully sent to Host 1 indi- cating that the file was infected and could not be sent to the receiving agent (see List 6). This denies message was received in Host 1 as shown in the Host 1 console (see List 5). The database was successfully updated with the information about the worm which included the worm's name, the worm's signature, the location and type of damage and the fileID.

- Verification of results

The researcher was able to further verify the detection using the five parameters (see Table 5).

Table V: Verification of Results 3

Name	Location of replica- tion and damage	Type of Replication	Damage type	File Status
WormProlaco (Worm3)	ScrapInsertedObj.ex	e 1	Create files	Bad

The researcher was successfully able to further verify the damage using the time stamp; for example, when (C://yaz//ScrapInsertObject.exe) was damaged, the properties shows the file was modified creating the new file (C://yaz//ScrapOutputObject.exe) and the time stamp revealed the time of this damage (see Figure 6).

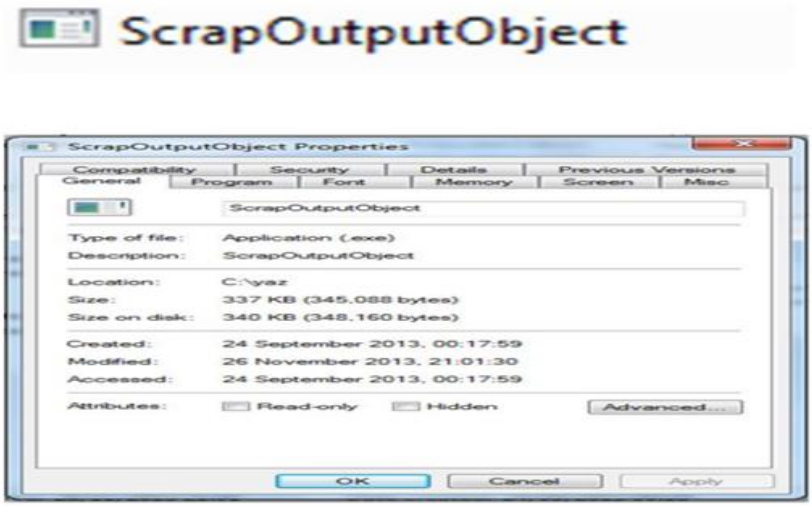


Figure VI: Verification of Results Worm 3

V. IV Experiment IV

Experiment 4 is conducted to determine if the system can detect a worm in a file that is being sent from Host 1 to Host 3 via DH and the Object Handler. Like the previous experiments, Experiment 4 aims to demonstrate that the system allows replication and damage to occur in the Dummy Host and that detection takes place using that damage. However, the difference with the previous experiments is that the type of replication was type 2 and the damage was to destroy the winload.exe as opposed to creating a new .exe file as shown in experiment 3. Moreover, the experiment will test if the overall system functions are working properly and that the consoles are displaying the correct information including reply and deny messages and file destination. Finally, the experiment also aims to check that updating the database with information about the detected worm is successful.

The expected outcomes are that the system is able to detect a worm by allow- ing the worm to replicate and cause damage in the Dummy Host. Moreover, it is expected that the host consoles will show the relevant information related to sending and receiving of files and information related to detection such as deny messages.

- **Results for Host 1 Console 4**

Experiment 4 was run by sending a file from Host 1 to DH. As with previous experiments, the Host 1 console shows the name of the file being sent, confirmation that the file was sent to DH and the reply from DH which indicates the status of the file? Note that in this case a worm was detected in the DH and the reply message is shown in the console as 'Answer - winload.exe file had been created from DHfile4' in the last line of the console showing the type of damage (see List 7).

Listing VII: Host 1 Console 4

1 Hello. My name is host1file4
2 the File change 0
3 the Agent id (agent -identifier :name DHfile4@146 .227.115.149:1099/ JADE)
4 the conversation ID DptBlaster1
5 the message had been sent
6 this is called from the DptBlaster2
7 == Answer <- winload.exe file had been created from DHfile4@146 .227.115.149:1099/ JADE

- **Results for DH Console 4**

The file is sent to DH for detection and the following console was shown (seeList 8).

Listing VIII: DH Console 4

1 Hello. My name is DHfile4
2 ********** This is before the replication Start *********
3 Found files for hashing:
4 -- c:\yaz\ ScrapInsertedObj .exe
5 -- c:\yaz\Test.txt
6 -- c:\yaz\cdrom.sys
7 -- c:\yaz\cmd.exe
8 -- c:\yaz\ originalcdrom .sys
9 -- c:\yaz\ originalcmd .exe
10 -- c:\yaz\ originalwinload .exe
11 -- c:\yaz\winload.exe
12 Hash value for the directory: bb4ef003fdb85f8b66bb95c2c2c2d80e
13 the winload.exe
14 ********** This is After the replication Start *********
15 Found files for hashing:
16 -- c:\yaz\ DetectionHandlerOutput .txt
17 -- c:\yaz\ ScrapInsertedObj .exe
18 -- c:\yaz\Test.txt
19 -- c:\yaz\cdrom.sys
20 -- c:\yaz\cmd.exe
21 -- c:\yaz\ originalcdrom .sys
22 -- c:\yaz\ originalcmd .exe
23 -- c:\yaz\ originalwinload .exe
24 -- c:\yaz\winload.exe
25 Hash value for the directory: bb4ef003fdb85f8b66bb95c2c2c2d80e
26 Database Connection successfully made
27 the SqlQuery Insert into FilesInfo (id , name , signature , locationOfTypeDamage) values (0,'File4 ', 'this2winload .exe ','This is about winload.exe ');
28 C:\ yaz0\winload.exeWritten
29 C:\ yaz1\winload.exeWritten the winload.exe file replicated 2 times in the different directory wit
30 the winload.exe file replicated 2 times
31 the winload.exe file is false positive
32 the signature of the file is this2winload .exe
33 the file is affected true
34 9 BBF91A3028052ACE1FD6E65E3B15ACA7E144FC2
35 586 E642F75B776CE9847FD5CCA69A9BD3A21B936
36 File Has changed
37 Original File Hash value 9 BBF91A3028052ACE1FD6E65E3B15ACA7E144FC2
38 Current file Hex value 586 E642F75B776CE9847FD5CCA69A9BD3A21B936
39 the Reply to the Sender (agent -identifier :name host1file4@146 .227.115.149:1099/ JADE :addresses (sequence http ://146.227.115.149:7778/ acc))

The DH console shows that the file is successfully received by the DH. More- over, the console shows that damage occurred in DH and a worm was successfully detected by detecting the damage that it caused. This infection was revealed by detecting a change in the Hash directory. Specifically, the console showed that the worm corrupted the winload executable file in (C://yaz//winload.exe) by replicating the file using the same name, however, in a different directory, namely (C://yaz0//winload.exe) and (C://yaz1//winload.exe), thus the damage type was Type 2. This caused significant damage because the operating system became corrupted.
Subsequently, a deny message is sent to Host 1 that the file was infected and can- not be sent to the receiving host (Host 3). The database was successfully updated with the information about the worm which included worm name, worm signature, type and location of damage and fileID. All of these actions are carried out success- fully by the system and indicated in the DH console.

- **Verification of Results 4**

Table VI: Verification of Results 4

Name	Location of damage or replication	Type of Replication	Damage type	File Status
Blaster (Worm 4)	C://yaz//winload.exe	2	Corrupts the win- load.exe	Bad

V. V Experiment V

The main aim of Experiment 5 is to check that the detection system is able to detect a worm through replication behaviour and damage in the Dummy Host. More specifically, the experiment will see if the Dummy Host allows replication and damage to take place and detection takes place using that replication or damage. Moreover, as with previous experiments, this experiment aims to check that the functions of the system such as communication and the sending and receiving of files and messages are working properly.
The expected outcomes of the experiment are that the system is able to detect a worm by allowing the worm to cause damage in the Dummy Host. Further expected outcomes are that the host consoles display the relevant information about sending and receiving of files and messages.

- **Results for Host 1 console 5**

The console for Host 1 clearly shows the name of the file being sent and confirmation that it has been sent to DH. Moreover, the console shows that Host 1 successfully received a reply message from the DH indicating the status of the file. In this experiment, the reply message shows that the file was infected, specifically, the worm damaged the 'cdrom.sys' (see List 9). The replication type was Type 1.

Listing IX: Host 1 Console 5

```
1 Hello. My name is host1file5
2 the Agent id ( agent -identifier :name DHfile5@146 .227.115.149:1099/ JADE )
3  the conversation ID DptEmailWorm1
4  the File change 0
5  the message had been sent
6  this is called from the DptEmailWorm2
7 == Answer <- The cdrom.sys corrupted from DHfile5@146 .227.115.149:1099/ JADE
```

- **Results for DH Console 5**

The file is sent to DH for detection and below is the console for DH which shows the actions that took place.

Listing X: DH Console 5

```
1  This is for the agent 5
 2 ********** This is before the replication Start *********
3 Found files for hashing:
4 -- c:\yaz\ DetectionHandlerOutput .txt
5 -- c:\yaz\ ScrapInsertedObj .exe
6 -- c:\yaz\Test.txt
7 -- c:\yaz\cdrom.sys
8 -- c:\yaz\cmd.exe
9 -- c:\yaz\ originalcdrom .sys
10 -- c:\yaz\ originalcmd .exe
11 -- c:\yaz\ originalwinload .exe
12 -- c:\yaz\winload.exe
13 Hash value for the directory: c8cd0412f924bc411dc421e89096507f
14  the output file Name C:\yaz\cdRom.sys
15 ********** This is After the replication Start *********
16 Found files for hashing:
```

17 -- c:\yaz\ DetectionHandlerOutput .txt
18 -- c:\yaz\ ScrapInsertedObj .exe
19 -- c:\yaz\TaskMon.exe
20 -- c:\yaz\Test.txt
21 -- c:\yaz\cdrom.sys
22 -- c:\yaz\cmd.exe
23 -- c:\yaz\ originalcdrom .sys
24 -- c:\yaz\ originalcmd .exe
25 -- c:\yaz\ originalwinload .exe
26 -- c:\yaz\winload.exe
27 Hash value for the directory: 276 ff61e4837cbbde0fdd9251abb191e
28 Database Connection successfully made
29 the SqlQuery Insert into FilesInfo (id , name , signature , locationOfTypeDamage) values (0,'File5 ', 'this2cdrom.sys ','This is about cdrom.sys ');
30 the signature of the file is this2cdrom.sys
31 A80D103EECFE831B93C01F092ABCDDAE90BCCD6F
32 EC313C1043C509EDA822DD2528536F0EE0E6AA94
33 File Has changed
34 Original File Hash value A80D103EECFE831B93C01F092ABCDDAE90BCCD6F
35 Current file Hex value EC313C1043C509EDA822DD2528536F0EE0E6AA94
36 the message had been sent

The DH console (see List 10) shows that the file was successfully received by the DH. Moreover, the console shows that damage occurred, this was indicated by 'File has changed' in the DH console line number 33 in reference to a change in the Hash value of the CDrom file (see List 10) lines 34 and 35. Specifically, this worm corrupted the CDrom driver cdrom.sys causing it not to work. This was verified by a modification to the file shown in the system properties of the CDrom. In this experiment only damage was detected and not replication, therefore, there is the possibility that the damage was caused by malware other than a worm or that the damage was caused by a worm but the replication was not detected.

A deny message is sent to Host 1, indicated by 'reply to sender' (see Figure 7) that the file was infected and could not be sent to the receiving agent. The database was successfully updated with the name of the worm, worm signature, location and type of damage and the fileID.

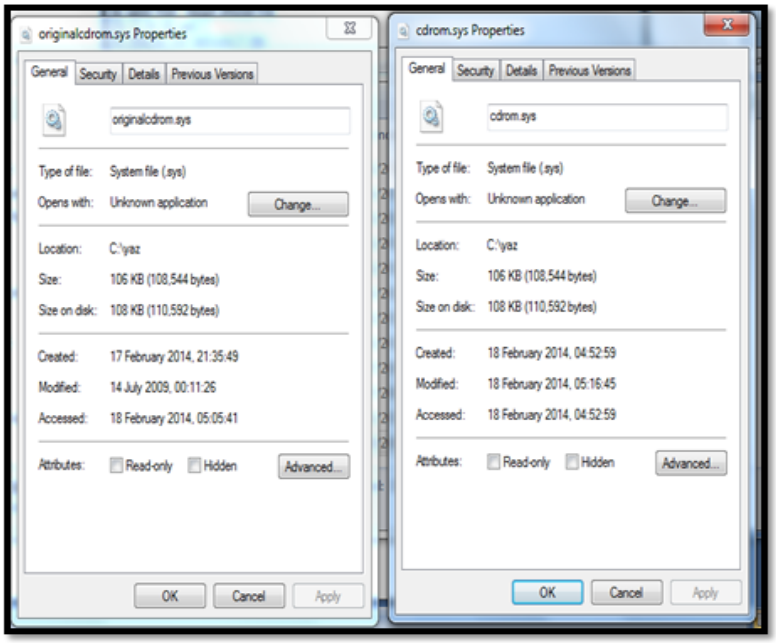


Figure VII: Verification of Results Console 5

Table VII: Verification of Results 5

Name	Location of damage or replication	Type of Replication	Damage type	File Status
EmailWorm (Worm 5)	cdrom.sys	none	Corrupts cdrom.sys	bad

V. VI Experiment VI

The main purpose of Experiment 6 is to send a file from Host 1 to Host 3 via the DH to determine if successful detection of a worm takes place. It is also an aim of the experiment to demonstrate that the system allows replication behaviour and damage to occur in the Dummy Host and is able to detect a worm through that replication behaviour or damage. Moreover, the experiment will determine if the system functions are working properly and that the consoles are displaying the correct information including reply and deny messages and file destination. The experiment also aims to check that updating the database with information about the detected worm is successful.

An expected outcome is that the detection system will detect a worm successfully through detecting replication behaviour or damage. Another expected outcome is that the consoles clearly illustrate the sending and receiving of a file and subsequent information upon detection such as deny messages.

- **Results for Host 1 Console 6**

The name of the file being sent and confirmation that it has been sent to the DH are successfully indicated in the Host 1 console (see List 11). Because subsequent detection in the DH found a worm, the console showed that Host 1 successfully received a reply message from the DH indicating that the file has been infected.

Listing XI: Host 1 Console 6

1 Hello. My name is host1Unknownfile
2 the File Name c:\\ yaz \\ test.txt
3 the File change 0
4 the Agent id (agent -identifier :name DHUnknownfile@146 .227.115.149:1099/ JADE)
5 the conversation ID DptUnknown1
6 the message had been sent
7 this is called from the DptUnknown2
8 == Answer <- The test.txt file had been increase the size by Unknownfile from DHUnknownfile@146 .22

- **Result for DH 2 Console 6**

The DH console shows that the file was successfully received by DH. Replication behaviour and damage occurred and an unknown worm was successfully detected through this replication and damage. The DH console showed that the file (C://yaz//test.txt) increased in size through replication by a worm. This increase in size was detected by a change in the Hash directory indicating that there was a change in one or more of the files in this folder, here the Hashing provides a list of files in this folder and it is through a change in a file name that reveals the issue. In this case a change was revealed through the output (C://yaz//testOutput.txt) indi- cating that replication of this file had taken place in the same directory, therefore, the replication type was Type 1. As a result, a reply message was sent to Host 1 indicating that the file was damaged and therefore, infected (see List 12). The database was successfully updated with the name of the worm, worm signature, location and type of damage and the file ID.

Listing XII: DH Console 6

1 Hello. My name is DHUnknownfile
2 ********** This is before the replication Start *********
3 Found files for hashing:
4 -- c:\yaz\ ScrapInsertedObj .exe
5 -- c:\yaz\Test.txt
6 -- c:\yaz\cdrom.sys
7 -- c:\yaz\cmd.exe
8 -- c:\yaz\ originalcdrom .sys
9 -- c:\yaz\ originalcmd .exe
10 -- c:\yaz\ originalwinload .exe
11 -- c:\yaz\winload.exe
12 Hash value for the directory: cfe17ac960f7b4cac223f120152620a7
13 the testOutput.txt
14 ********** This is After the replication Start *********
15 Found files for hashing:
16 -- c:\yaz\ ScrapInsertedObj .exe
17 -- c:\yaz\Test.txt
18 -- c:\yaz\cdrom.sys
19 -- c:\yaz\cmd.exe
20 -- c:\yaz\ originalcdrom .sys
21 -- c:\yaz\ originalcmd .exe

22 -- c:\yaz\ originalwinload .exe
23 -- c:\yaz\testOutput.txt
24 -- c:\yaz\ testOutput0 .txt
25 -- c:\yaz\ testOutput1 .txt
26 -- c:\yaz\winload.exe
27 Hash value for the directory: 071102 d3c480e8cd130a63179ffc2c8b
28 Database Connection successfully made
29 the SqlQuery Insert into FilesInfo (id , name , signature , locationOfTypeDamage) values (0,'unKnown 'testOutput.txt ','This is about testOutput.txt ');
30 the unknown File increased the size
31 the signature of the file is testOutput.txt
32 7486 DA833344A5DD4EDEAE3BCE83D9C7DDC5517C
33 D294C929FCF5D46AB8656B8E5E806A5119D4E372
34 File Has changed
35 Original File Hash value 7486 DA833344A5DD4EDEAE3BCE83D9C7DDC5517C
36 Current file Hex value D294C929FCF5D46AB8656B8E5E806A5119D4E372
37 the Reply to the Sender (agent -identifier :name host1Unknownfile@146 .227.115.149:1099/ JADE :addresses (sequence http ://146.227.115.149:7778/ acc))

- **Verification of Results 6**

Table VIII: Verification of Results 6

Name	Location of damage or replication	Type of Replication	Damage type	File Status
Unknown worm	C://yaz//test.txt	1	Increase size of file	Bad

V. VII Experiment VII

The previous experiments were carried out to establish if the WDS is capable of detecting a worm sent between hosts. Experiment 7 is concerned with determining if the WDS allows a file that is not infected to pass between hosts via the DH. Like with other experiments, experiment 7 is concerned with checking if all of the mechanisms of sending and receiving files and messages are functioning correctly and that the database update functions are working.

The expected outcome is that the WDS will search for a worm in a file using signature-based detection and through detecting replication behaviour and damage, and that the file will be found to be clean. Subsequently, the file will be allowed by the DH and the Object Handler to be received by the destination host.

- **Results for Host 1 Console 7**

The Host 1 console showed that a file (file7 which is cmd.exe file) is sent to the DH. In this experiment no replication behaviour or damage is detected and the file is determined to be uninfected. As a result the Host 1 console shows the reply message from the DH that the file was sent to the receiving host, Host 3.

Listing XIII: Host 1 Console 7

1 Hello. My name is host1file7
2 the File change 1
3 the Agent id (agent -identifier :name DHfile7@146 .227.115.149:1099/ JADE)
4 the message had been sent
5 == Answer <- Successfully received the good file from DHfile4@146 .227.115.149:1099/ JADE

- **Results for DH Console 7**

The DH console showed that the file was received and that no replication or dam- age took place indicated by the fact that there was no change in the Hash directory. Subsequently, the DH console displayed that there was no replication behaviour and no damage, indicated by 'the file had not changed' (see List 14)

Listing XV: DH Console 7

1 Hello. My name is DHfile7
2 ********** This is before the replication Start *********
3 found files for hashing:
4 - c:\yaz\ ScrapInsertedObj .exe
5 - c:\yaz\cdrom.sys
6 - c:\yaz\cmd.exe
7 - c:\yaz\ originalcdrom .sys
8 - c:\yaz\ originalcmd .exe

9 - c:\yaz\ originalwinload .exe
10 - c:\yaz\test.txt
11 - c:\yaz\winload.exe
12 hash value for the directory: 0 ada7922844c78b597bbf6a0891445dc
13 ********** This is After the replication Start *********
14 found files for hashing:
15 - c:\yaz\ ScrapInsertedObj .exe
16 - c:\yaz\cdrom.sys
17 - c:\yaz\cmd.exe
18 - c:\yaz\ originalcdrom .sys
19 - c:\yaz\ originalcmd .exe
20 - c:\yaz\ originalwinload .exe
21 - c:\yaz\test.txt
22 - c:\yaz\winload.exe
23 hash value for the directory: 0 ada7922844c78b597bbf6a0891445dc
24 the file is affected false
25 9 BBF91A3028052ACE1FD6E65E3B15ACA7E144FC2
26 9 BBF91A3028052ACE1FD6E65E3B15ACA7E144FC2
27 File Had not changed
28 the message had been sent

- **Results for Host 3 Console 7**

Because no worm was detected the file can be sent to the destination host, Host 3. The Host 3 console shows that the file was successfully received from DH (see List 15).

Listing XVI: Host 3 Console 7

1 Hello. My name is host3file7
2 I received the good file
3 == Answer <- the message successfully received from DHfile7@146 .227.115.149:1099/ JADE

- **Verification of Results**

Table IX: Verification of Results 7

Name	Location of damage or replication	Type of Replication	Damage Type	File Status
Good File	None	None	None	Good

VI. System Evaluation

In order to evaluate the system the above experiments were conducted using a number of different files or messages that were either clean or infected. The results of each experiment were presented in the previous section. These results form the basis of the evaluation of the system that is presented here.

V. VII Experiment VII

Before the experiments were conducted a number of evaluation criteria were established in order to conduct the experiments, here these criteria are used to assess the success of the system in light of the results of the experiments. Specifically, the criteria are as follows:

• Dummy Host allows replication behaviour and damage
• Dummy Host detects replication behaviour and damage
• Detecting unknown worms
• Clean files successfully sent between hosts
• Successful updating in database

- **Replication Behaviour and Damage in Dummy Host**

One of the criteria was that the WDS allows replication behaviour and damage to take place in the Dummy Host. For experiments 1,2,3,4 and 6 where the files were infected by a worm, replication or damage successfully took place in the Dummy Host. Specifically, in experiments 1,2,3,4 and 6 replication behaviour and damage was detected, in experiment 5 only damage occurred and in experiment 7 there was no replication or damage.

- **Detection through Replication Behaviour and Damage in Dummy Host**

The main system criterion was that the WDS can detect a worm through replication behaviour or damage that takes place in the Dummy Host. The WDS successfully detected both known and unknown worms through replication behaviour and dam- ages. All the experiments demonstrated that the detection through using the Hash directory and Hash value for known worms and Hash directory for unknowns was successful.

- **Unknown Worms**

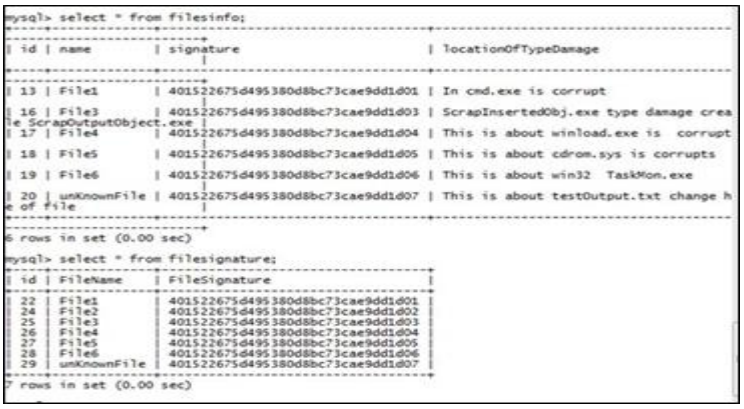
The experiments were concerned with checking whether or not the WDS could detect replication behaviour and damage of worms. However, it was also important to specifically check if the WDS was capable of detecting the replication behaviour and damages of unknown worms, the reason for this is that an unknown worm may skip detection by replication behaviour and would therefore, be detected by damage and the experiment had to evaluate this.

- **The Good File**

The system was able to show in experiment 7 that it was capable of identifying a good file, i.e., a file without a worm. When the system was run, no signatures were matched, no anomaly behavior was found and no damage took place in the Dummy Host and the system was able to indicate good file. For other detection systems that use signature and behavior based detection mechanisms there is the possibility of falsely identifying the presence of a worm. In the experiment it was shown that there was no change in the Hash directory when a clean file was sent. Any change in the Hash directory would indicate that damage or replication is taking place, if the Hash directory remains unchanged then there is no worm engaged in damage or replication present.

- **Updating Database**

Information about the detected worms were successfully updated in the database in two different tables, the first table is the 'files info' table which was updated with information which included the name, the signature and location and type of damage, the second table 'File signatures' was updated with the file name and signature. The database was successfully updated for experiments 1, 2,3,4,5 and 6 where a worm was detected; this is shown in Figure 8.



```
mysql> select * from filesinfo;
+----+-------------+----------------------------------+-----------------------------------------------------
| id | name        | signature                        | locationOfTypeDamage
+----+-------------+----------------------------------+-----------------------------------------------------
| 13 | File1       | 401522675d495380d8bc73cae9dd1d01 | In cmd.exe is corrupt
| 16 | File3       | 401522675d495380d8bc73cae9dd1d03 | ScrapInsertedObj.exe type damage crea
e ScrapOutputObject.exe |
| 17 | File4       | 401522675d495380d8bc73cae9dd1d04 | This is about winload.exe is  corrupt
| 18 | File5       | 401522675d495380d8bc73cae9dd1d05 | This is about cdrom.sys is corrupts
| 19 | File6       | 401522675d495380d8bc73cae9dd1d06 | This is about win32  TaskMon.exe
| 20 | unKnownFile | 401522675d495380d8bc73cae9dd1d07 | This is about testOutput.txt change h
e of file
+----+-------------+----------------------------------+-----------------------------------------------------
6 rows in set (0.00 sec)

mysql> select * from filesignature;
+----+-------------+----------------------------------+
| id | FileName    | FileSignature                    |
+----+-------------+----------------------------------+
| 22 | File1       | 401522675d495380d8bc73cae9dd1d01 |
| 24 | File2       | 401522675d495380d8bc73cae9dd1d02 |
| 25 | File3       | 401522675d495380d8bc73cae9dd1d03 |
| 26 | File4       | 401522675d495380d8bc73cae9dd1d04 |
| 27 | File5       | 401522675d495380d8bc73cae9dd1d05 |
| 28 | File6       | 401522675d495380d8bc73cae9dd1d06 |
| 29 | unKnownFile | 401522675d495380d8bc73cae9dd1d07 |
+----+-------------+----------------------------------+
7 rows in set (0.00 sec)
```

Figure VIII: Update Databases

VII. Conclusion

This paper has presented the experimentation and evaluation of the detection system. The reasons for the experiment and the evaluation criteria of the system were explained. The results were presented for each of the successive experiments for known worms, unknown worms and good files and it was shown how the required functionality, i.e., communication, detection and updating, performed. The results were further verified by the researcher and used to provide an evaluation of the system. The experiments showed that actual damage took place in the Dummy Host which was a major system requirement. The evaluation showed that the system was successfully able to detect worms through detecting replication behaviour and damage, and also damage alone without replication, for the latter we consider that this could be caused by unknown worms or other malware. It was important to conduct the experiment for unknown worms as well as known worms to show that the system could detect them through damage alone if necessary. Moreover, the experimentation showed that the system was successfully able to detect known worms through signature-based detection.

Future work could include evaluating the performance of the WDS, specifically through evaluation of the performance of the hash directory and hash value in the system directory. Additionally, future work could include evaluation of the WDS in terms of its precision and recall efficiency. Dummy hosts or virtual machines are widely used to analyse worm behaviour, and one of the issues that developers have to contend with is that the authors of malware are increasingly finding ways of identifying virtual machines from real ones. Therefore, future work could include devising a way of hiding the dummy host so that malware creators are not able to derive information that will help them to create more resistant and hard- to-detect worms. Because the WDS uses damage to detect worms, future work could include the possibility of using the WDS to detect other malware such as viruses. Virtual machines are already been used for the analysis of different types of malware and the extension from analysis to detection, a contribution of this study, could be used for all malware types.

References

Journal Papers:

[1] Y. Alsaawy and S. AL Amro, Design and Implementation of Computer Worms Based on Monitoring Replication and Damage, *IJCSNS International Journal of Computer Science and Network Security, 16(11)*, November 2016

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

[3] M. Shankarpani, K. Kancherla, R. Movva, and S. Mukkamala. Computational intelligent techniques and similarity measures for malware classification. *In Computational Intelligence for Privacy and Security*, pages 215–236. Springer, 2012.

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

[5] F. El-moussa and A. Jones. Malware analysis: The art of detecting malicious activities. *In Proceedings of the 7th European Conference on Information Warfare*, page 51. Academic Conferences Limited, 2008.

[6] L. Zeltser. Analyzing malicious software. *In CyberForensics*, pages 59–83. Springer, 2010.

[7] M. F. Zolkipli and A. Jantan. A framework for defining malware behavior using run time analysis and resource monitoring. *In Software Engineering and Computer Systems*, pages 199–209. Springer, 2011.

[8] N. Idika and A. P. Mathur. *A survey of malware detection techniques*. Purdue University, page 48, 2007.

[9] J. C. Rabek, R. I. Khazan, S. M. Lewandowski, and R. K. Cunningham. *Detection of injected, dynamically generated, and obfuscated malicious code.*

[10] K. Rieck, T. Holz, C. Willems, P. Du¨ssel, and P. Laskov. *Learning and classification of malware behavior. Detection of Intrusions and Malware, and Vulnerability* Assessment, pages 108–125, 2008.

[11] M. Dalla Preda, R. Giacobazzi, S. Debray, K. Coogan, and G. M. Townsend. Modelling metamorphism by abstract interpretation. *In Static Analysis*, pages 218–235. Springer, 2011.

[12] Q. Zhang. *Polymorphic and metamorphic malware detection*. doctoral diss, North Carolina State University, 2008.

[13] P. Beaucamps, I. Gnaedig, and J.-Y. *Marion. Behavior Analysis of Malware by Rewriting-based Abstraction* - Extended Version. Rapport de recherche, May 2011.

[14] B. B. H. Kang and A. Srivastava. Dynamic malware analysis. *In Encyclopedia of Cryptography and Security*, pages 367–368. Springer, 2011.

[15] L. Cavallaro, P. Saxena, and R. Sekar. On the limits of information flow techniques for malware analysis and containment. *In Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 143–163. Springer, 2008.

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

[17] M. Christodorescu and S. Jha. Static analysis of executables to detect mali- cious patterns. Technical report, DTIC Document, 2006.

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

[19] U. Bayer, A. Moser, C. Kruegel, and E. Kirda. Dynamic analysis of malicious code. *Journal in Computer Virology, 2(1):*67–77, 2006.

[20] X. D´efago, F. Petit, and V. Villain. Stabilization, Safety, and Security of Dis- tributed Systems: 1*3th International Symposium, SSS 2011*, Grenoble, France, October 10-12, 2011, Proceedings, volume 6976. Springer, 2011.

[21] K. Dunham. *Mobile malware attacks and defense*. Syngress, 2008.

[22] M. D. Preda. *Code Obfuscation and Malware Detection by Abstract Interpretation*. doctoral diss, Universit'a degli Studi di Verona Dipartimento di Informatica, 2007.

[23] A. Chaudhuri, P. Naldurg, and S. Rajamani. A type system for data-flow integrity on windows vista. *ACM Sigplan Notices*, 43(12):9–20, 2009.

[24] P. Royal, M. Halpin, D. Dagon, R. Edmonds, and W. Lee. Polyunpack: Automating the hidden-code extraction of unpack-executing malware. *In Computer Security Applications Conference*, 2006. ACSAC'06. 22nd Annual, pages 289–300. IEEE, 2006.D. Spinellis. Reliable identification of bounded-length viruses is np-complete. Information Theory, IEEE Transactions on, 49(1):280–284, 2003.

[25] M. Anbar, S. Manickam, A.-S. Hosam, K.-S. Chai, M. Baklizi, and A. Almomani. Behaviour based worm detection and signature automation. *Journal of Computer Science, 7(11)*, 2011.