

Honeypot Technique For Detection of Malware Signatures

Harpreet kaur*, Shivam kumawat

Lovely Faculty of Technology and Sciences

Lovely Professional University, Phagwara, Punjab

harpreet.23521@lpu.co.in, shivamkumawat45@gmail.com

ABSTRACT- Use of USB drives for transferring digital information is very common and which also a main sources wide variety of malwares to the system. A privately owned smart device that is connected to a corporate LAN network can be a potential USB channel to malware infections. For example, airspace systems are the safest places to store important data. However, unlike network connections, there is no authentication or filtering by USB connection streams. Therefore, a deliberate or un-intentional transmission of a USB device or a mobile device infected with malware in the airspace is sufficient to transfer the infections to those systems. It becomes necessary to identify and analyse such malicious intrusions in order to advance restrictive measures. Honeypots/Honeynets are the devices that support this task, they collect sample for analysing malwares. We provide a new sort of honeypot which it finds malware by mimicking the removable device's software and copying it to the virtual device itself.

Keywords- *honeypot, malware, Intrusion detection, USB drives.*

INTRODUCTION- Malware is a risk in today's digital world. Internet allows malware to spread rapidly around the world, and there are other ways to distribute it. In 2011, McAfee's antivirus company identified more than 20 million new unique malware displays. That is why it is important to fight against malware by developing ever more complex defence systems Web technology is very popular for distributing malware on USB sticks..

In this paper, we are dealing with malware - malware that does not work properly on computers that do not belong to the perpetrator tries to gain political or economic gain. While computer security has steadily improved, malware users are not only a major threat to mainstream users, but also to government agencies and the Internet as a whole. If your computer has entered malicious software, you may need to act even if this malicious code wants it. For example, a machine that affects it can damage harmful personal information[1]. For analysis of malware cases, automated systems that attempt to display an infected computer or a weak desktop are often used. Honey pots often work on conventional systems. With virus scanners and anti-phishing hackers in networks, companies can create new types of malware for analysis. It is therefore important to first develop a unique signature for

unknown malware. The Internet exists in many different ways, but the fundamental problem is in every way how malware affects the network. Hackers often track networks that are vulnerable to network services, and even the entire operating system, to detect infections on the network. This approach has successfully attracted malware to expose a weak device on the Internet with useful information. Collecting data from inbound network traffic also makes it impossible to have an estimate of number of systems infected with specific type of malware.

Literature Review- [2]Honeywell Industrial USB Threat Report, in this report from Honeywell things about malware have been discussed in detail such as what they actually are, what they do when they get into a system, how many types of malware are there, how they propagate but most importantly how they spread through USB and what can be done to stop it. Conclusion of this report is that the types of threats discovered on inbound USB removable storage were more serious than the research team anticipated, the overall amount of malware was relatively small. The most important findings point to the inevitability of USB threat exposure, with nearly half of the SMX gateways analyzed blocking at least one malicious file. When so many of the threats discovered are targeting ICS and potentially disruptive, every threat needs to be prevented. This report shares Honeywell USB security research findings to advance industry dialogue and threat prevention collaboration, in hopes of lowering cyber attack risk to industrial operations worldwide

[3] Shadow honeypots is hybrid architecture that consists of characteristics of honeypots and Intrusion detection system. The type of intrusion detection system is anomaly detection system. In this approach, group of anomaly detectors is used to monitoring and classification of traffic to protected network. Using shadow honeypot, various types of attacks are detected prior to get infect the protected network.

[4] Bait and switch honeypot like snort, Intrusion detection system redirects the traffic sent by attacker to a production network. When the attack is detected successfully, packets of first attack are dropped and remaining all the traffic from attacker is diverted to honeypot host. Hence, attack is analysed without letting attacker to know about the process and protecting production system from such intrusions.

[5] Honeyd is a low interaction honeypot which is installed on UNIX system and supports UDP,TCP[6] and ICMP protocols. This type of honeypot responds according to the services (TCP,UDP and ICMP) running on virtual environment. Packet is modified before response packet is sent to the network by honeypot's personality engine called dispatcher to match

with the behaviour of network of configured OS personality. [7] Honeycomb is improvised version of honeyd. All the intended traffic which is directed to honeypot is considered to be malicious. This approach will result in more number of false positives even for non-malicious traffic. New signatures[8] are created for all incoming traffic to honeypot and updation of existing signatures is done in case similar traffic is detected.

PROBLEM FORMULATION- In this world where malwares are on their verge and are getting smarter in ways that it is really difficult to detect them and counter them.

Malwares have been propagating not only through network connection or internet but also through USB devices[9] and we underrate the destruction caused by USB spread malwares we propose a system that helps in gathering information about those malwares specifically so that a further improved defence mechanism can be founded using those Information.

We propose a system that acts as a honeypot for those malwares it comprises of two main parts:-

- 1) A driver that acts as a removable device and convinces the operating system that a removable device is injected. As malwares propagate when they discover a removable device they try to write on the removable device which is an indication of getting infected.
- 2) A bus that just works as a storage capacity and it's motive is to collect data from that malware that is trying to infect the removable device that we have made using the driver.

The goal of this honeypot is to assemble information, such as information about the malware itself, author information, or malware-related infections[10]. In our case, the campaign focuses on knowledge of these honeypot infections and examples of the malware program. During the life of the malware, such technique can try to collect copies of executable malware (.exe) libraries. In this framework, the term "life cycle" states to a process by which malware infects a virtual machine, hides in the system, performs unwanted(malicious) actions, such as passing information to remote clients or reading logs, and lastly infecting others virtual machines. Reproduction, that is, infection by other machines, is the stage at which most of the honeypots are involved in the spread of malware. One reason is that distribution is a common feature of this type of malware. However, there is still a lot of work to be done to spread malware across the network, while other media are receiving very little attention.

IMPLEMENTATION- After revising the basic theoretical principles of the new idea, investigation is done of how the suggested scheme can be applied. Windows XP is chosen as the target system for the execution. Regardless of its version, it is broadly being used and is targeted by various types of malware. Still, the code can also be developed with little effort to run on other various versions of Windows. The implementation of a virtual USB stick is to create a virtual storage (virtual device) supported by the image file. Applications must be able to read and write data on the device and at the same time transfer all these I / O functions to an image file that our storage stores. Secondly, the removable device must look accurately as the portable storage device for all applications that request information about it. This is where the available solutions do not offer sufficient flexibility: tools such as FileDisk mimic the hard drive, but cannot make it give the impression as a removable device. The implementation of a honeypot can be complex at times enough to provide such utilities, but we do not require the proposed system to be more intricate than required. That is why there are different approaches to achieve the above two main goals: simulate storage and make it portable.

Microsoft offers simulation of USB devices, allows you to execute user level codes and route them to the kernel with custom drivers. However, the main objective of the framework seems to be to test the pilot. This also requires a very weak version of the simulator, which is not necessary for the concept presented here. Looking at the software options to consider when a new mobile storage is connected, the operating system sends the message to registered applications. The receiving application can display the other message parameters to see if a new connected device is a portable storage device. This method is used by Conficker[11] and Stuxnet[12]. Our virtual appliances must be implemented so that malware can detect them with both methods. Before we consider creating a portable virtual device in this regard, we need to develop device processing in the Windows kernel[13]. The devices are signified by the device parts and each object is controlled by a device driver. When connecting new devices, Windows determines which drivers to download from the newly connected device driver device driver. The loaded driver can then create a device object itself and assign it a specific device ID. Again, the driver of this device is loaded based on identification, etc. Thus, the device components and associated drivers are stacked on top of each other. Each driver processes queries directed to the peripheral portion and then passes them to the driver of the next lower

device object. On the lower layers of the storage device driver key, we usually find the corresponding bus drivers (for example, IDE, USB), somewhere above the disk driver (general driver for all types of storage) and associated systems, and above, file systems. to allow users of the room app to access the device. Our goal is to create portable devices for the operating system. It seems that the concept of capacity reduction in Windows is determined by the hard drive. It determines whether a storage device is classified as portable and determines whether the device views the device as a portable device. The Tooldrive.sys driver replaces the disk drive driver of the displayed USB flash drive and ensures that it is reported as a portable storage device for the top-level drivers. From user space, the device looks a lot like a truly portable storage device. Of course, a careful inspection of the storage device stack may reveal the true nature of the USB drive, but this confirmation requires much more effort than the simple User API settings described above.

1) You can register devices at the root. That is, a device with a specified ID is assumed to be present at system startup and the required drivers are loaded accordingly. The PCI bus driver is an example of a driver loaded in this way.

2) The devices can be listed in other parts of the device (bus list). If the driver of an existing device creates a new device, Windows loads the appropriate driver. It is clear that such a series must begin somewhere where the number of roots is needed. Examples of drivers loaded by bus list are the USB drivers (initially registered by the PCI bus driver to which the USB controller is connected) and the disk class driver (registered by the corresponding bus driver). which storage device is connected). The virtual USB drive can be connected as desired. We can not accept root counting when the driver is loaded at startup.

METHODOLOGY

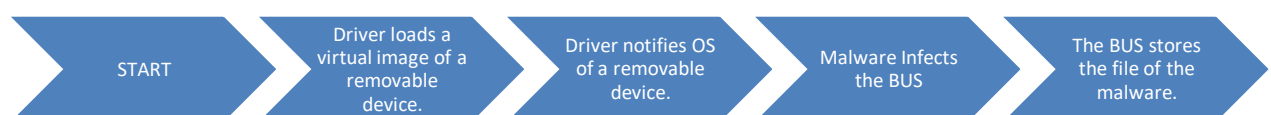


Figure 1: Methodology of the proposed work

RESULT AND ANALYSIS- After analysing, we will estimate the recommended idea based on the flowchart presented above. Entire calculations were performed on a virtual machine on which Windows XP is installed, which was reset after each test case run by replacing the hard disk contents with a set binary image.

The main focus for this assessment was to examine the working of concept and to test whether the assumed malware models infects the USB drive and time taken by it to infect the system.

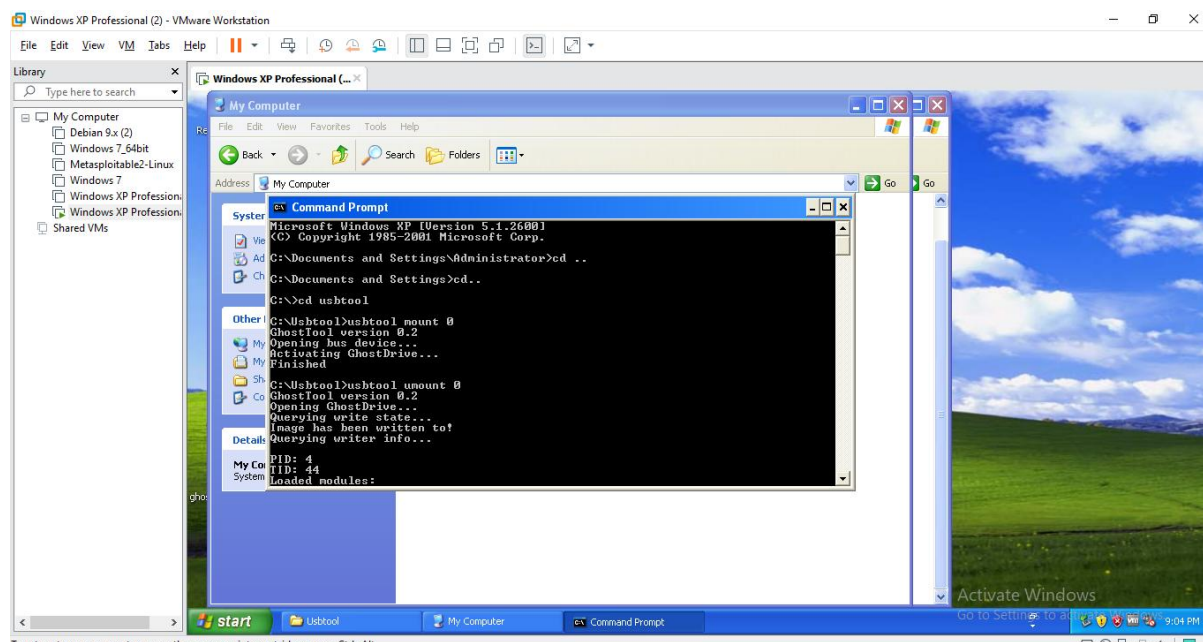


Figure 2: creation of flash drive

In figure 2, it will create a virtual flash drive and a .img.in the C drive to store the collected data in the virtual drive. Then, it will take some time for malware to copy itself in the flash drive.

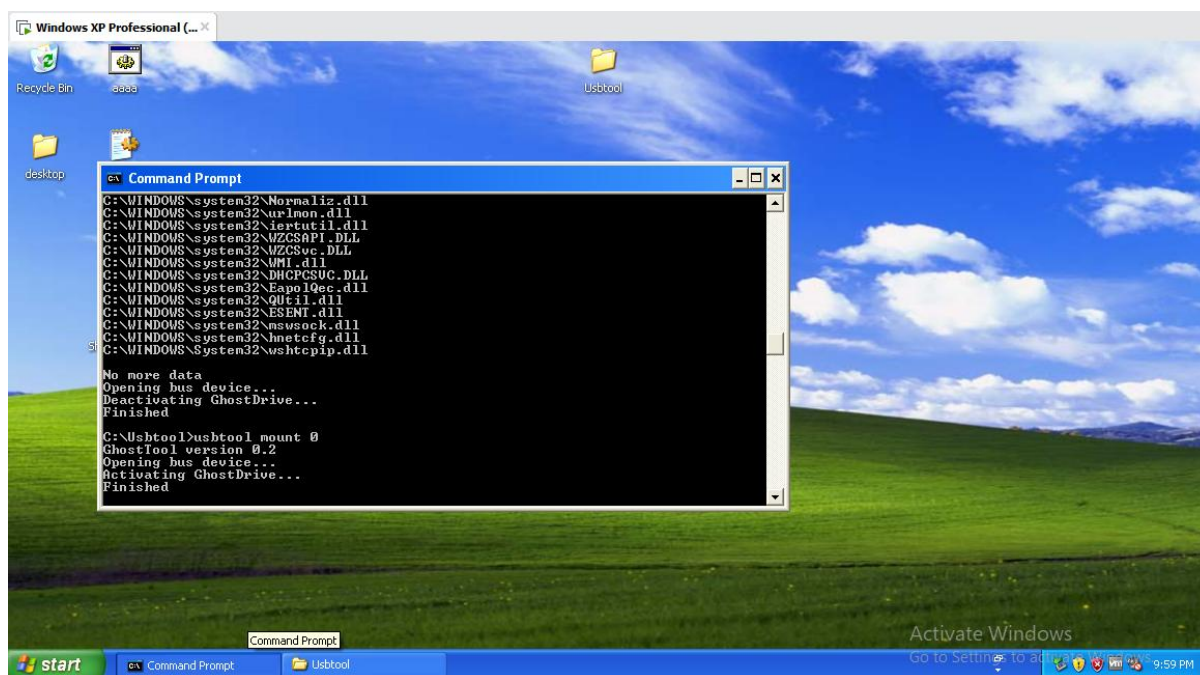


Figure 3: Comparison of malware signatures

In figure 2, Usbtool mount 0 command is used which will eject the virtual usb drive and it will compare the signature of any virus. If the signatures of any existing virus is found, then the presence of malware is reflected in the virtual drive. Further, stored malware in .IMG file can be used for investigation.

TESTING AND CONCLUSION

The malware models used for the tests were carefully chosen for a basic function. The specimen must consistently infect the USB device connected to the infected engine. Ever since, this is the simple statement that the honeypot is the target of the malicious program, we must resist all test samples.

We have selected two examples of Conficker malware in the test suite, as they are fairly advanced malicious programs.

Failure to detect the honey pot may also mean that existing high-level (in technical complexity) malware can be deceived into infecting the virtual USB key.

Table I samples and evaluation given by different virus scanners[14]

SET OF TEST SAMPLES M_T WITH RESPECTIVE CLASSIFICATION BY VIRUS SCANNERS.

Title	Classification by virus scanners		
	Kaspersky	Symantec	Microsoft
Sample A	Worm.Win32.AutoIt.r	W32.Badday.A	Worm:Win32/Yuner.A
Sample B	Worm.Win32.AutoRun.rwp	W32.Ircbrute	Worm:Win32/Hamweq.G
Sample C	Worm.Win32.AutoRun.nuh	Trojan.Packed.NsAnti	Worm:Win32/Taterf.B
Sample D	Worm.Win32.AutoRun.lkw	W32.Baki.C	Worm:Win32/Autorun.DX
Sample E	Worm.Win32.AutoRun.zki	W32.SillyFDC	Backdoor:Win32/Darkshell.A
Sample F	Virus.Win32.VB.ki	W32.Mikbaland	Worm:Win32/Autorun.CA
Conficker I	Net-Worm.Win32.Kido.ih	W32.Downadup.B	Worm:Win32/Conficker.B
Conficker II	Net-Worm.Win32.Kido.ih	W32.Downadup.B	Worm:Win32/Conficker.C

One may observe that Stuxnet, although often mentioned in this paper, is not part of the set. This is because it is impossible to reliably infect this particular malware, even with a physical USB drive.

Symantec's analysis[15] suggests that modifying the malware configuration file is sufficient to infect removable media infection.

However, this method did not work for us and we could not directly call the corresponding routines in the malware dll file.

The DLL[16] is stored only in coded form and inserted into selected processes at the beginning of the system. Therefore, we do not add Stuxnet to the test kit. However, if Symantec describes a routine Stuxnet infection, malware cannot distinguish between a USB drive and our virtual equivalent.

This project at the time of testing has been implemented on a virtual machine created on VMware workstation. The host operating system that has been used in testing was Windows XP .It has been tested on various malware samples and has been able to detect the malwares in expected manner and in limited amount of time. First we have to build the code for this we use the window development kit (WDK).

REFERENCES

- [1] P. A. Norberg, D. R. Horne, and D. A. Horne, "The privacy paradox: Personal information disclosure intentions versus behaviors," *J. Consum. Aff.*, vol. 41, no. 1, pp. 100–126, 2007.
- [2] U. S. Bus, "Honeywell Industrial USB Threat Report," 2018.
- [3] K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, and a D. Keromytis, "Accuracy Shadow Honeypot Honeypot Anomaly Detection Scope," *Symp. A Q. J. Mod. Foreign Lit.*, pp. 129–144, 2005.

- [4] J. W. and A. Gonzalez, "The Bait and Switch Honeypot." [Online]. Available: <http://baitnswitch.sourceforge.net/>.
- [5] N. Provos, "Honeyd: A Virtual Honeypot Daemon," *Proc. 10th DFNCERT Work.*, pp. 1–7, 2003.
- [6] S. Bishop, M. Fairbairn, M. Norrish, P. Sewell, M. Smith, and K. Wansbrough, "Rigorous specification and conformance testing techniques for network protocols, as applied to TCP, UDP, and sockets," *Comput. Commun. Rev.*, vol. 35, no. 4, pp. 265–276, 2005.
- [7] C. Saadi and H. Chaoui, "Cloud Computing Security Using IDS-AM-Clust, Honeyd, Honeywall and Honeycomb," *Procedia Comput. Sci.*, vol. 85, no. Cms, pp. 433–442, 2016.
- [8] V. S. Sathyanarayan, P. Kohli, and B. Bruhadeshwar, "Signature Generation and Detection of Malware Families," in *Information Security and Privacy*, 2008.
- [9] H. Carvey and C. Altheide, "Tracking USB storage: Analysis of windows artifacts generated by USB storage devices," *Digit. Investig.*, vol. 2, no. 2, pp. 94–100, 2005.
- [10] G. Zhao, K. Xu, L. Xu, and B. Wu, "Detecting APT malware infections based on malicious DNS and traffic analysis," *IEEE Access*, vol. 3, pp. 1132–1142, 2015.
- [11] N. Fitzgibbon and M. Wood, "Conficker.C A Technical Analysis," *Analysis*, pp. 1–16, 2009.
- [12] A. Thabet, "Stuxnet Malware Analysis Paper," p. 29, 2011.
- [13] A. Ganapathi, V. Ganapathi, and D. Patterson, "Windows XP Kernel Crash Analysis," in *Proceedings of the 20th Conference on Large Installation System Administration*, 2006.
- [14] S. Poeplau, J. Gassen, and E. Gerhards-Padilla, "A honeypot for arbitrary malware on USB storage devices," in *7th International Conference on Risks and Security of Internet and Systems, CRIStIS 2012*, 2012.
- [15] C. Seifert, R. Steenson, I. Welch, P. Komisarczuk, and B. Endicott-Popovsky, "Capture - A behavioral analysis tool for applications and documents," *Digit. Investig.*, vol. 4, no. SUPPL., pp. 23–30, 2007.
- [16] M. N. A. Khan, "Performance analysis of Bayesian networks and neural networks in classification of file system activities," *Comput. Secur.*, vol. 31, no. 4, pp. 391–401, 2012.