Pervasive Malware Propagation Mechanism and Mitigation Techniques

Amit Kumar
Master of Engineering,
Student,
Department of Computer Engineering,
K.J Somaiya College of Engineering,Vidyavihar,
Mumbai
Maharashtra,India

Pallavi Kulkarni
Assistant Professor,
Department of Computer Engineering,
K.J Somaiya College of
Engineering,Vidyavihar,
Mumbai,
Maharashtra,India

ABSTRACT
Malwares i.e. malicious code/softwares poses prevalent threat to businesses and network across distributed systems. Like it is said in order to catch criminals, we have to think like a criminal, likewise in order to catch cyber criminals/terrorists, we have to think like a cyber-criminal. Malware campaigns have been the driving engines for cyber-warfare being used by cyber criminals & black hat hackers to target organizations, various governments, and financial institutions for leverage & selfish profits, since early decade. In the recent trends of past years, the sophistication of malware campaigns have grown more complex to perform targeted successful attacks and bypass the prevailing & evolving defense mechanisms out there. Our approach is motivated by the factor that malwares breed on the vulnerability of the software applications running across the web. Idea behind pervasive malware propagation mechanism is to provide insight towards various exploitable scenarios based on vulnerabilities and software coding flaws in the software system, its architecture and over the network. Understanding the control flow structure of malware propagation into the system & over the network provides greater insight into how vulnerabilities are being exploited , how target surface identification is being carried out by the attackers, how exactly the exploits are being delivered using the payloads & what mechanism is being used to maintain the access to the exploited victim over the network. Eventually some suggestions as precautionary mitigation mechanisms to stop the malware propagation.

General Terms
Malware Propagation Mechanism, Information Security Threat Analysis.

Keywords
Pervasive, Malware Propagation Mechanism, Vulnerability, Exploitation, Threat Surfaces, Vulnerability Identification, Payload , Software Flaw ,Coding Flaw, Vulnerability Patching, Exploit kit, botnet , command & control center , attack surfaces, Malicious code, Advanced attack.

1. INTRODUCTION
Malwares [6][8][9] poses prevalent threat to businesses and network across distributed systems. Like it is said in order to catch criminals, we have to think like a criminal in order to catch cyber criminals/terrorists, we have to think like a cyber-criminal. Therefore an understanding of the advanced malware propagation mechanism which is prevalent in this ever changing internet scenario, becomes of extreme necessity.

Software and systems complexity can have a profound impact on information security. Such level of sophistication is not only imposed by the imperative technical challenges of constantly monitored distributed systems & networks, but also through the advancements in exploits and malware spreading mechanisms determined by the underground economics. In addition, operational business restraints (disruptions and consequences, manpower, and end internet user's satisfaction, increases the sophistication of the problem domain that security analysts must adequately operate within. Implementation of efficient response is highly dependent over the strict time duration involved in the discovery of a particular vulnerability & the patching time of the vulnerability, in order to assess & minimize the impact over the business.

With known and unknown vulnerabilities getting updated on an exponential basis the need of the hour is to understand the propagation mechanism of the most radical malware threats and map them according to their threat surfaces on their respective target operating environment.After successful mapping of the operating environment variables of the known threat surfaces, it's important to take our stance on the unknown threats by devising mechanism to identify those threats. Since any malware out in the wild is crafted only for a targeted objective with multiple backup options available to sustain in an ever changing environment. Efforts will be made to ascertain the variables and parameters which helps the malwares to sustain and evade modern threat prevention mechanisms.

2. PREVAILING RESEARCH SCENARIOS
Current research trends show extensive use of targeted malware campaigns being launched to target companies & governments to steal financial & other sensitive information [10][11]. Malwares are malicious software programs deployed by cyber attackers to compromise computer systems by exploiting their security vulnerabilities. Motivated by extraordinary financial or political rewards, malware owners are exhausting their energy to compromise as many

networked computers [10] as they can in order to achieve their malicious goals. A compromised computer is called a bot, and all bots compromised by a malware form a botnet [8][9]. Botnets have become the attack engine of cyber attackers, and they pose critical challenges to cyber defenders. In order to fight against cyber criminals, it is important for defenders to understand malware behavior, such as propagation or membership recruitment patterns, the size of botnets, and distribution of bots. Till date, we do not have a solid understanding about the size and distribution of malware or botnets [1]. Researchers have employed various methods to measure the size of botnets, such as botnet infiltration, DNS redirection, and external information [1][5]. These efforts indicate that the size of botnets varies from millions to a few thousand [7][8]. There are no dominant principles to explain these variations. As a result, researchers desperately desire effective models and explanations for the chaos. Researchers have revealed that time zone has an obvious impact on the number of available bots [1]. Also indicated that network topology has an important impact on malware spreading through their rigorous mathematical analysis [1]. A fundamental characteristic of malwares is self-propagation, i.e., a malware can infect vulnerable hosts and use infected hosts to self-disseminate. A key component of malware propagation is malware-scanning methods, i.e. how effectively the malware finds vulnerable targets [2]. For attackers, an open question is how certain information can help them design fast-spreading malwares. The information can be from coarse to fine, including the number of vulnerable hosts, a distribution of vulnerable hosts, the locations of detection systems, and individual vulnerable hosts. This work focuses on aggregated information, i.e. vulnerable-host distributions .The vulnerable-host distributions have been observed to be bursty and spatially inhomogeneous

3. SYSTEM DESIGN

Considering the fact that malwares are prevalent across network, we will try and focus our research implementation on the parameters which are crucial for malware modelling & propagation incorporated by an expert botnet master or a malware writer. In this paper we will try to apply a modelling technique commonly used for epidemic & population dynamics modeling to predict the spread of a malware. Current research work implementation lab setup is done in a virtual lab environment with Kali Linux machine as server that is command & control (C&C) for the virtualized network environment. Simulation for the attack on three kinds of threat surfaces have been targeted, which are the most prevalent on the radar of any malware writer and serve as a vulnerable target based on its increasing level of difficulty. Following threat surfaces [5][7] have been identified and targeted on the Windows7 machine in a controlled virtualized environment, namely:

1) Microsoft Office Word 2007-Buffer Overflow vulnerability exploitation
2) Adobe Reader Image Embedded with External URL
3) Browser based exploitation example on Internet Explorer

However the scope of applications can vary according to the choice of most persistent tools being used by malware writers and also the most popular operating system platforms like Android mobile platforms, Linux machines, Windows mobile platforms etc. can be targeted accordingly [3][4][5]. As we are well aware that the above mentioned target applications have widespread use across the industry and among standalone users, which happens to be the primary reason to choose these target applications for my experimentation purpose. Security researchers have developed exploits for these kinds of vulnerabilities & these exploits are publicly available, which can be customized according to the need and demand of an attacker and type of suited malware campaign. Now creation and development of an exploit requires a lot of effort and high level of expertise in shell code scripting and other low level programming knowledge. With the guidance and support from security researchers across the security community it was possible to successfully craft an undetectable exploit code for Microsoft Office Word 2007 document which exploits the vulnerability of buffer overflow vulnerability in Microsoft Office Word 2007 applications on any Windows XP & Windows 7 & machine, for our experimentation it's performed on Windows 7 operated machines.

Targeting specific set of different vulnerabilities in the three different applications as mentioned above, to show & demonstrate how exactly a malware propagates itself by exploiting any vulnerability across the network for any particular target application. Based on the limitations & dependencies of the particular malwares targeting these kind of vulnerabilities and statistically will try to model a mathematical equation for the spread mechanism of malware modeling based on the Lotka-Volterra equations [2], which are widely used for modeling population dynamics of species competing for the same resources. Based on our analysis the success rate and failure rates for the malware have been set based on our calculated parameters. Since our focus is on the propagation mechanisms of a malware modeling based on Lotka-Volterra equations [2], following assumptions are applied for this implementation model:

1) Infected computers cannot infect others directly.
2) All vulnerable computers are equally likely to be infected.
3) There is only one infection per computer.
4) Probabilities of infection are given by a Poisson distribution.
5) One malware entity per C&C server.
6) Total number of computers remains the same.

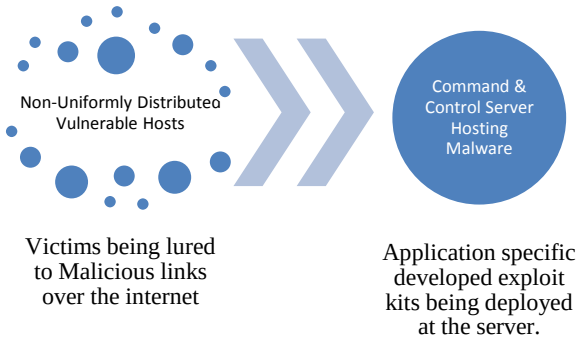

Figure 1. Showing General Model of Malware Propagation Mechanisms

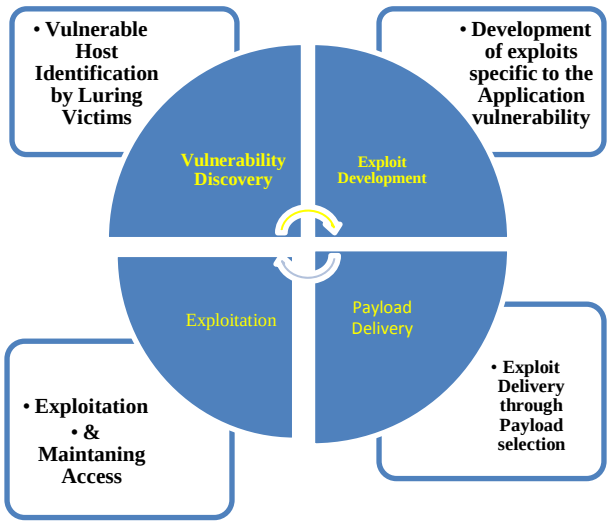

Figure 2: Showing General Control Flow Structure of Parameters affecting Malware propagation

3.1 Onto Malware

To model the spread of malware we will make use of similar paradigms of Lotka-Volterra equations [2] but design it a little differently to reflect the nature of malware proliferation. Specifically, simply using our knowledge of how malware can spread by exploiting certain set of known and unknown vulnerabilities to design equations that allow us to model and analyze the dynamics of its proliferation [3][5]. The model consists of a network of virtual computers that periodically talk to each other. The network can be a host based internal network or just some computers that share information via USB drives. Vulnerable computers can be infected with 2 types of malware – known malware and so-called 0-Day malware [4][6][8]. The distinction here is that Anti-Virus companies have signatures for known malware. Once a computer is infected and that infection is detected, it can be quarantined. Here quarantined means that a system-admin/network admin has to isolate it from the rest of the network. After the computer is quarantined it is disinfected and then put back on the network. Antivirus companies periodically update their signatures, so computers infected with unidentified malware eventually get converted to an infection with identified known malware, i.e. malware they were infected with can become "known". Malware writers [3][5] are more likely to write malware if there are a large number of known-vulnerable computers and some number of active malware already in the wild. This way the effect of each malware would be amplified and they could improve upon extant malware. Malwares periodically stop working due to several factors and bad domain/IP lists are periodically updated to block C&C server traffic. 0-Day malware is harder to write, but it is probably written better and has a higher chance of succeeding.

4. MALWARE PROPAGATION MODEL IMPLEMENTATION

In this experiment we have simulated and created exploitation scenario for three types of most widely used applications across the industry, namely:

1) Microsoft Word Document 2007- Buffer overflow vulnerability exploitation

In this application vulnerability exploitation scenario targeted application is the most popular office document application which is being used worldwide across industries. Targeted buffer-overflow vulnerability which exists in the popular Microsoft Office 2007 word software. Here, it's shown how vulnerability is exploited over the network using document sharing tools. Exploitation in real time scenario involves non-uniformly distributed vulnerable hosts. Specially crafted exploit code have been developed using python code & shell code for the purpose of proof of concept to exploit the existing vulnerability of buffer overflow inside the application. Propagation of the infected/malicious word document have been done using conventional methods through email [4][6], hyper link creation and luring the victims to download the file, spreading and posting the links over social media .

Following are the screenshots of the process followed to propagate the malicious word document through specially crafted exploit code in detectable & undetectable scenarios:

In the following screenshots coverage of the exploitation process has been included, proof of concept is written in python code aligned with the shell-code to overflow the buffer inside a Microsoft Office word -2007 document & call any arbitrary exe file or any malicious code of attacker's choice to be executed

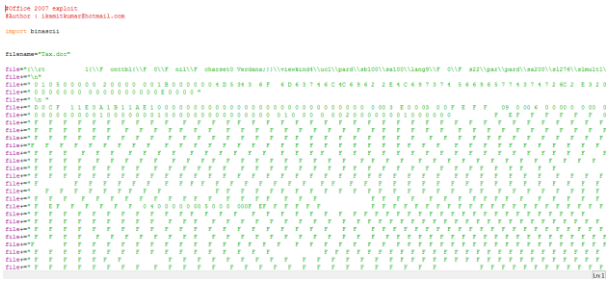

Figure 3: Random buffer shell-code

Buffer shell code created to overwrite the instruction pointer inside the word application. Crafted exploit shell code will provide us with the highest privilege inside windows machine. This crafted code directly runs in the memory of the stack.

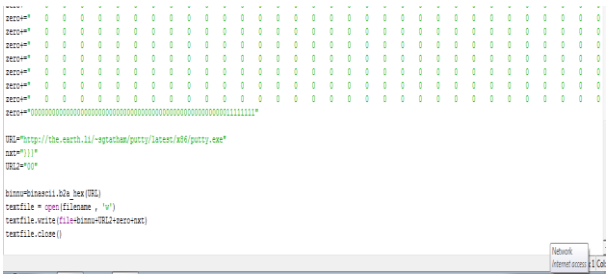

Figure 4: Python code for adding the malicious/targeted URL executable

Python code has been used to generate the proof of concept for the vulnerability exploitation of word document as per required by an attacker.

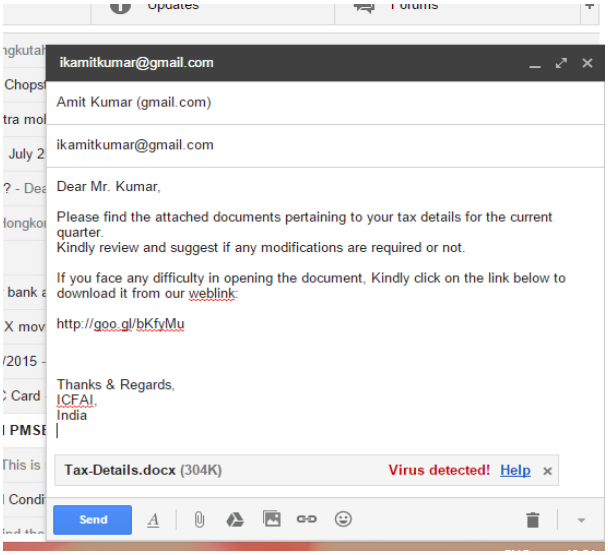

Figure 5: Error message alert when the exploit code gets detected

Virus alert message being displayed over the network while attaching it on email. This shows that exploit code needs to be updated or some variations are required such that it can be attached over email successfully without raising any suspicion.

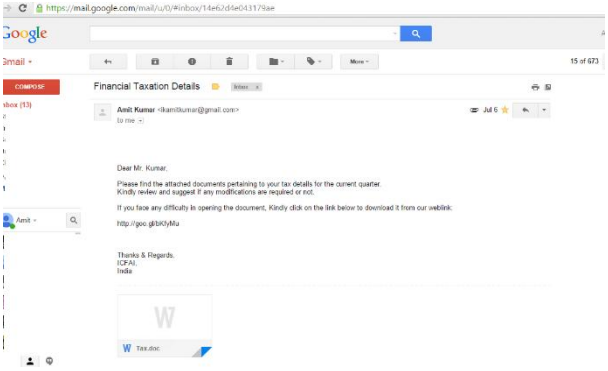

Figure 6. Successful attachment over email

In the above figure successful attachment of malicious word file has been made, after making the necessary customization to make the word exploit code undetectable.

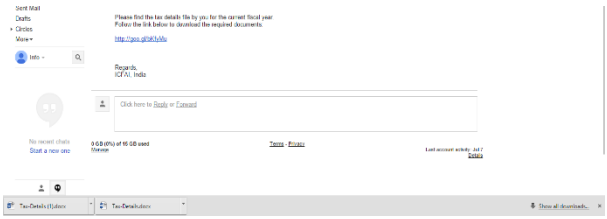

Figure 7. Target Host downloading the attachment.

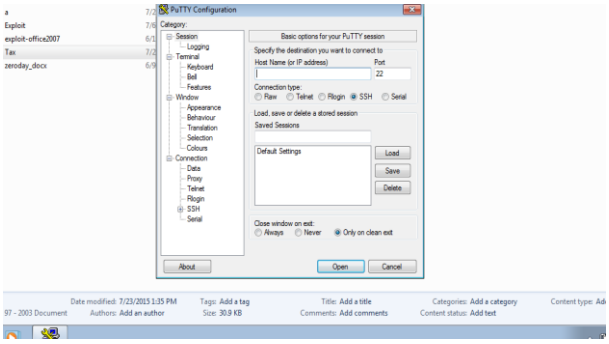

Figure 8: The execution of an arbitrary Executable

In our case, it was putty.exe after the user tries to open the downloaded word document.

2) Browser based exploitation

In this application vulnerability exploitation scenario, target is the most popular inbuilt browser Internet Explorer which comes pre-installed in systems. Targeted vulnerability in the LNK process and will make use of WebDAV to run the exploit [8][9] using Metasploit framework to exploit the vulnerability [8][9]. Metasploit framework is an open source tool which is being used extensively as the most popular tool of choice for the security experts and researchers across the security community. Exploitation [8] in real time scenario involves non-uniformly distributed vulnerable hosts.

In the following screenshots it covers, the process of exploitation of Internet Explorer using the open source security tool of Metasploit framework:

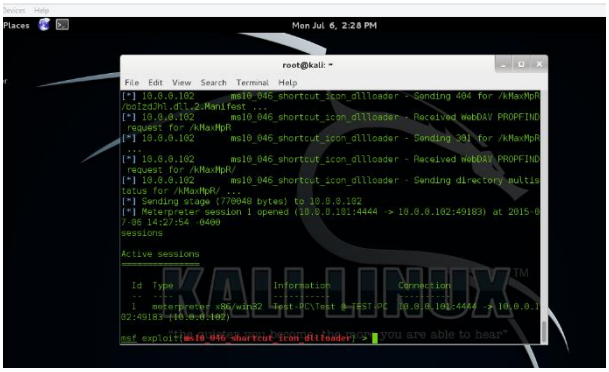

Figure 9: Exploitation & reverse shell creation

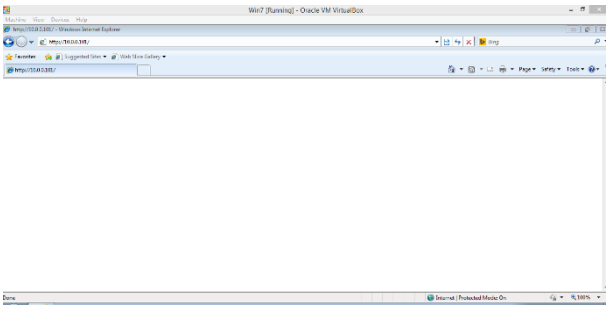

Figure 10: Internet Explorer Browser activity on target machine running Windows 7.

Figure 11: Captured screenshots of desktop information about Windows 7 machine inside Metasploit framework.

Figure 12: Directory listing of the compromised Windows 7 machine using Metasploit framework.

Figure 13: Network connection statistics of the compromised Windows 7 machine using Metasploit framework.

Figure 14: The command prompt takeover of the compromised Windows 7 machine using Metasploit.

Figure 15: The critical system information of the compromised Windows 7 machine using the Metasploit.

3) Adobe Reader Image Embedded with External URL

PDF today is already a mature, broadly supported and universally accepted electronic document format. We will be using a social engineering attack of embedding an external URL inside an image embedded in Adobe PDF reader such that when the user tries to open the PDF file every time it will be redirected to the embedded URL inside it. Exploitation [8] in real time scenario involves non-uniformly distributed vulnerable hosts.

Following screenshots show the process of URL embedding inside the embedded image inside a PDF file:

Figure 16: The Embedded the link for the Executable file inside PDF.

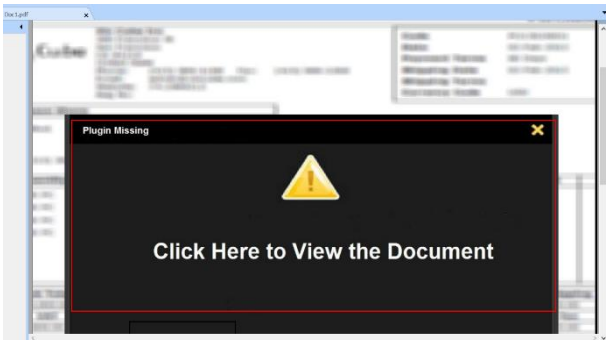

Figure 17: The Error Message Image inside the PDF

It will help to lure target victim to click on the embedded link.

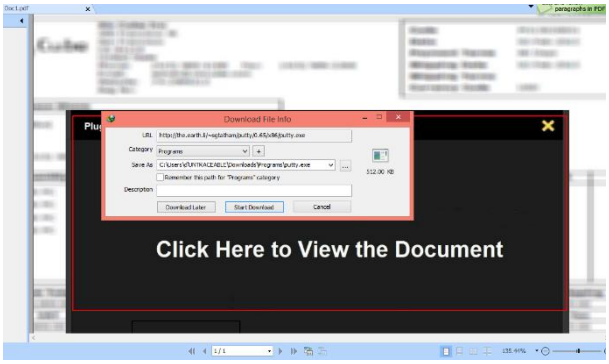

Figure 18: The executable link in our case putty.exe

Ready to be downloaded executable, once the victim clicks on the embedded link over the image.

5. ANALYSIS OF MALWARE PROPAGATION MODEL PARAMETERS

Using the Lotka-Volterra equations from biomathematics [2] to depict the malware [8][9] propagation model:

Let x(t) be the vulnerable target computers and y(t) be the malware population at any given time instant t. Then, the following differential equations model the malware-vulnerability interaction:

$$\frac{dx}{dt} = r.x(t) - a.x(t).y(t)$$

$$\frac{dy}{dt} = b.x(t).y(t)$$

$$\big(x(0), y(0)\big) = (x_0, y_0)$$

where,

$\boldsymbol{x_0}$ is the initial number of vulnerable host ,

$\boldsymbol{y_0}$ is the initial number of malware [8] population

r is the rate of infection

a is the rate of malware successful propagation,

b is the rate of malware[8] failure.

The equation have no analytical solutions, but are instead approximated numerically.

Based on our experiments as shown in the screen shots for the creation of exploit [8] code to target the buffer overflow vulnerability [8][9] inside the word application, it can be observed that there are a lot of dependencies that are involved ,which the attacker might face as a challenge, while crafting the shell code in order to avoid detection. The crafted shell-code and the python code are used to create the proof of concept for the exploitation of the vulnerability [8][9] of buffer overflow inside word application. Our target executable is putty.exe which is completely harmless software, which we have used for our demonstration purpose. Our objective here is to show how exploit codes specially crafted according to any particular vulnerability [9] specific to any application can be deployed by malware writers and what are the difficulties & challenges faced at different level of this multi-staged attack. These kind of applications are targeted by the malware writers in a multi-staged attack where the malware server hosts the exploit-kit containing the exploits for the specific target based on the type of application to be targeted and lure the victim to download or click on a link or an embedded image as shown in our experimentation.

6. STRATEGIES FOR MITIGATION MECHANISM

Exploitation [8] in real time scenario involves non-uniformly distributed vulnerable hosts. The base line to mitigate any vulnerability from the core of any application would be to create and develop securely coded application right from the beginning of the application development cycle. Gradually implementing secure coding practices would be a great help in mitigating vulnerabilities which result out of coding flaws inside the applications. Mitigation of any vulnerability is a periodic process. New vulnerability getting discovered and older vulnerabilities getting exploited in the wild shows the significance of keeping the applications updated as soon as any new update is released. However, the time period in which any application vulnerability is discovered and the patch release time duration may depend upon the respective vendors of the different applications. Like in this case Microsoft releases vulnerability patches every Tuesday for its applications and software across the globe likewise various other vendors and companies have their own cycle of releasing the patches. Now during this duration the risk of exploitation of any vulnerability for the specific application is very high. Careful and informed usage of any applications can avert any kind of malware attacks provided the user takes the necessary precautions like for example in our case, the user could easily avert the PDF attack by getting suspicious about the clicking over the error message inside the PDF file. There are several vendor websites informing the users about the do's and don'ts and have made a set of rules and guidelines on their website portal for the users to be followed which might help them to keep safe to certain extent. However the risk is always there. So mitigation factors might work or it might fail because there are no perfect systems and there are no perfect malwares.

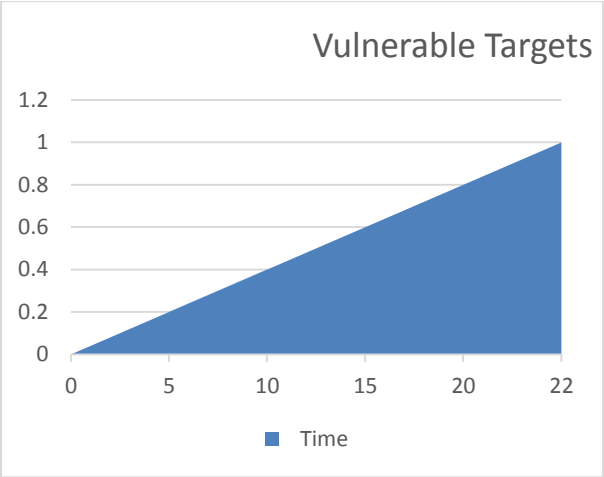

Fig 19: General Trend showing fraction of vulnerable Targets versus Malware Propagation Time period (Days)

7. CONCLUSION

The experiments & results demonstrated in this paper have certain limitations although when creating the model we made a number of assumptions that may not always be true, these results can still be useful. The detection of the exploit code is time dependent i.e. it might be possible that when the actual experiment was being conducted the exploit code was undetectable, but after sometime due to unforeseen updates across the algorithms of email malware detection systems, it might start getting detected in due course of time. Although we have conducted the experiment for one known vulnerability and made the exploit code undetectable, it may be possible that may unknown vulnerabilities are still existing inside the application which needs to be discovered or may have been discovered and patched in due course of time. We have conducted the experiments in a totally controlled environment inside a virtual lab consisting of Kali Linux machine and Windows 7 machine as per with the required tools and applications. Through our experiment we have tried to show the glimpse of the parameters which play pivotal role in success and failure of any malware propagation. The concept of making a known vulnerability exploitable with modification in the shell-code to make it undetectable provides insight about how malwares and exploit kits target the host population to sustain in this ever changing scenario over the internet. Although the conducted experiment has shown one known vulnerability and made the exploit code undetectable, it may be possible that may unknown vulnerabilities are still existing inside the application which needs to be discovered or may have been discovered and patched in due course of time. However, when creating the model a number of assumptions have been made that may not always be true, these results can still be useful, especially if the model is fit to real data. Small improvements in detection rate of known and 0-Day malware can make a big difference in the number of users subject to infection. For example, a 9% increase in the immediate detection can result in almost complete abolition of infected computer population, substantially decreasing the need for forensic analysis and potentially reducing required IT budget. Overall, quicker signature generation utilizing sandboxing techniques to detect

0-Days can have a profound impact in reducing the number of infected computers, thereby protecting user's data and preventing proliferation of botnets.

8. ACKNOWLEDGMENTS

Our thanks to the experts in the security research community who have contributed towards development of various sophisticated advanced mechanisms to study the behavioral analysis of the malwares and its propagation dynamics. We would like to show our gratitude towards the information security research community which supported us through out to guide us in our research & development process. Special thanks to H.D Moore the creator of Metasploit & Sandeep Kamble, Information Security Researcher, for their personalized guidance in the process of vulnerability exploitation techniques. We would also like to thank our college K.J Somaiya College of Engineering, Department of Computers, for this wonderful opportunity and providing the extensive support during the lab work. Last but not the least we are thankful to our family and friends for their continuous support.

9. REFERENCES

[1] Shui Yu, Song Guo , and Ivan Stojmenovic, Fool Me If You Can: Mimicking Attacks and Anti-attacks in Cyberspace, IEEE Transactions on Computers, 139-151 , http://dx.doi.org/10.1109/TC.2013.191

[2] Ajay Gupta, Daniel C. Du Varney, Computer Security Applications Conference, *2004. 20th Annual*, 116-125, 1063-9527 http://dx.doi.org/10.1109/CSAC.2004.47

[3] Kevin M. Carter, Nwokedi Idika, and William W. Streilein: Probabilistic Threat Propagation for Network Security IEEE Transactions on Information Forensics and Security, VOL. 9, NO. 9, September 2014**,** 1394 - 1405, 1556-6013, http://dx.doi.org/10.1109/TIFS.2014.2334272

[4] Seungwon Shin, Guofei Gu, , Narasimha Reddy ,and Christopher P. Lee: A Large-Scale Empirical Study of Conficker, IEEE Transactions on Information Forensics and Security , Volume 7 Issue 2, April 2012 Page 676-690 , 1556-6013, http://dx.doi.org/10.1109/TIFS.2011.2173486

[5] Zesheng Chen and Chuanyi Ji,:An Information-Theoretic View of Network-Aware Malware Attacks, IEEE Transactions on Information Forensics and Security, 30 June 2009, 530–541, 1556-6013, http://dx.doi.org/10.1109/TIFS.2009.2025847

[6] Sheng Wen, Wei Zhou, Jun Zhang,Yang Xiang, Wanlei Zhou: Modeling and Analysis on the Propagation Dynamics of Modern Email Malware, IEEE Transactions on Dependable and Secure Computing, 20 November 2013, 361 – 374, 1545-5971, http://dx.doi.org/10.1109/TDSC.2013.49

[7] Krishna K. Ramachandran and Biplab Sikdar: Dynamics of Malware Spread in Decentralized Peer-to-Peer Networks IEEE Transactions on Dependable and Secure Computing, 03 December 2010, 617 – 623, 1545-5971, http://dx.doi.org/10.1109/TDSC.2010.69

[8] Ping Wang, Sherri Sparks, and Cliff C. Zou,Member: An Advanced Hybrid Peer-to-Peer Botnet, IEEE Transactions on Dependable and Secure Computing, 18 July 2008, 113 – 127, 1545-5971, http://dx.doi.org/10.1109/TDSC.2008.35

[9] Silvio Cesare, Yang Xiang, and Wanlei Zhou: Malwise—An Effective and Efficient Classification System for Packed and Polymorphic Malware , IEEE Transactions on Computers, 19 March 2012, 1193 – 1206, 0018-9340, http://dx.doi.org/10.1109/TC.2012.65

[10] Ravishankar Borgaonkar: An Analysis of the Asprox Botnet, 2010 Fourth International Conference on Emerging Security Information, Systems and Technologies, July 18, 2010, 148-153, http://doi.ieeecomputersociety.org/10.1109/SECURWARE.2010.32.

[11] H. Binsalleeh , T. Ormerod , A. Boukhtouta , P. Sinha , A. Youssef , M. Debbabi , and L. Wang : On the Analysis of the Zeus Botnet Crimeware Toolkit, 2010 Eighth Annual International Conference on Privacy Security and Trust (PST), 17-19 Aug. 2010, 31 -38, http://dx.doi.org/10.1109/PST.2010.5593240.