International Journal of Future Generation Communication and Networking
Vol. 13, No. 1, (2020), pp. 1013-1023

Success Rates of Honeypots in P2P Botnet Tests

Meerah M. Al-Hakbani,[1] and Mostafa H. Dahshan[2]
[1] *Department of Computer Science, College of Computer and Information Science, King Saud University, Riyadh, Saudi Arabia.*
[2] *Department of Computer Engineering, College of Computer and Information Science, King Saud University, Riyadh, Saudi Arabia*
[1] *email: malhakbani@ksu.edu.sa*, [2] *email: mdahshan@ksu.edu.sa.*

Abstract

Network security is a critically important aspect in our lives and is vital in protecting our environment from attacks. Some security systems use honeypots as a defence method to monitor the behaviour of a botnet by attracting a botmaster to add it in their botnet. Additionally, some mechanisms have been proposed to help the botmaster differentiate between a honeypot and a real device. In this study, we developed a method to help security defenders perform the authentication procedure developed by the botmaster to prevent a honeypot from being a member of their botnet. The presented method uses a fake infection command on the hosts that will be infected during the authentication process. This research involves a simulation to evaluate the performance of the presented method. To increase the credibility of our outcomes, we simulate the ZeroAccess botnet by using the Monte Carlo method. We show that this method offers a better chance for honeypots to bypass the botnet defence.

Keywords: *botnet, P2P, peer to peer, honeypot, network, security.*

1. Introduction

Network security is important as it provides safety in the digital world. It helps businesses and organisations reduce the risk of data theft and protect their workstations from spyware. Distributed denial of service attacks and identity theft are examples of attacks caused by a botnet, which can impact millions of people and businesses [1].

Botnet is a common term used to define networks of infected hosts called bots. These infected hosts have bot software that enables a botmaster to control their systems remotely [2].

Botnets usually perform tasks that cannot be performed by a single computer. The work is initiated by a botmaster device. The device sends instructions to a small number of bots, and those bots propagate the instructions to other bots [3][1].

A peer-to-peer botnet is similar to a peer-to-peer network. When a new bot joins a botnet, it downloads the relevant bot software that contains a start-up list of peers. The peers in this list will be used to update the neighbouring peer information. this is called the bootstrap procedure. If security defenders obtain the initial peer list, they can shut down those peers and prevent the botnet from growing [4] [5].

Some botnet defence mechanisms that prevent botnet propagation have been proposed. F. Castaneda et al. [6] proposed a method that transforms a malicious worm into an anti-worm that then disinfects the original virus. The worm spreads itself using the same mechanism as the original worm. The architecture of the proposed method is divided into three stages. The first is a worm-detection stage that uses a honeypot-based system to capture and analyse the worm. The second stage is the design and implementation of a

ISSN: 2233-7857 IJFGCN

completely automated anti-worm generator. The third stage is an anti-worm propagation scheme. Researchers used a payload search algorithm to transform the message of the original worm into an anti-worm, and then embedded the generated anti-worm code in the payload. The anti-worm code is responsible for stopping all the activities generated by the existing worm as well as detecting and preventing new attacks against the hosting machine. However, the proposed anti-worms have several limitations. Some of them are: a worm could patch a system by removing the vulnerability that it uses to exploit the system. Network impact can limit the spread of the anti-worm. Additionally, the anti-worm needs to be generated quickly and spread at least as fast as the original worm, and moreover, it is illegal in many countries to impose a system that the user does not control.

F. Leder et al.. [7] developed methodologies that can be used to tackle the botnets from inside. These methodologies use a weak point in the infrastructure to manipulate, disrupt, or block. The methodologies are based on three different points of attack: taking down the C & C server, which is only possible if the botnet uses a centralised structure, location of the C & C server is known, and provider cooperates, redirecting malicious traffic to a sinkhole node to analyse it and drop it so that it cannot reach its original target, cleaning infected systems, and removing the installed bots. This is the most complex countermeasure. The system owners or administrators are responsible for cleaning infections from their systems. The strategies employed can be categorised as mitigation, manipulation, and exploitation. Mitigation strategies slow down the botnet by consuming its resources, for instance. Manipulation strategies can alter or remove some commands, dropping collected personal data or issuing commands to make bots stop doing some tasks. Lastly, exploitation strategies use bugs found in bots to perform actions on the infected machines.

C. Xiang et.al [8] proposed a technique that could mislead a malicious bot using its own propagation mechanisms to spread a BotSpoofer instead of spreading itself. This technique is called botnet spoofing. A BotSpoofer is a computer program that avoids an original attack and provides extra protection. The botnet spoofing technique targets persistent bots that carry at least one file in the hard disks (normally DLL or EXE). Note that there are also non-persistent malware types that exist in memory and disappear after the victimised computer is turned off. Therefore, botnet spoofing exploits the property of a persistent bot that obtains its file path for auto-start registration. Botnet spoofing tricks a bot into retrieving a fake path. This technique is independent of the vulnerability, protocol and structure of the botnet C & C.

Some security defenders use a honeypot in conjunction with other security tools to detect such a botnet. The honeypot is a computer system that is used to attract an attacker to add it so that valuable information about the purpose and activity of the attacker may be obtained [9]. However, some researchers developed methods to be used by botmasters to avoid adding honeypots in the botnet. These methods act as a test at the beginning of the infection to check whether or not the device is a honeypot [10].

This paper presents a new method to increase the chance to pass the honeypot detection procedure in the advanced two-stage reconnaissance worm (ATSRW) method developed in [10]. The presented method uses a fake infection command on the hosts that will be infected during the authentication process. This will help security defenders to add honeypots to the P2P botnet which help them detect and analyse botnet attacks.

The rest of the paper is organised as follows. Section II discusses the honeypot detection method introduced by the ATSRW and a way to bypass it. The proposed method is presented in Section III. Section IV provides an analysis of the presented method. Section V presents numerical results regarding the performance of the presented method. Finally, Section VI concludes the paper.

ISSN: 2233-7857 IJFGCN

2. Avoiding Honeypot Detection in Advance Botnet Attack

2.1. Detection Method (ATSRW)

Wang et.al in [10] [11] [12] designed a mechanism called the ATSRW to detect a honeypot in P2P botnet. This mechanism has two main components:

A peer list and a spearhead code responsible for compromising a vulnerable computer.

A main-force code that uses an authorisation key to allow a new bot to join the constructed botnet.

The procedure to add a host in the botnet is illustrated in Figure 1. After the vulnerable host B is infected by the spearhead code, it will attempt to infect arbitrary hosts. If C is one of the arbitrary hosts, host B registers the IP address of host C, and continues finding other hosts to infect. After that, host C sends the infection tuple <B's IP, C's IP> to the hosts in B's peer list using a Transmission Control Protocol (TCP) connection.

Suppose host P is a bot that passed the honeypot detection test and is one of the hosts in B's peer list, then host P signs the IP address of host C and sends it to host B. Then, host B checks if the received tuple is signed by a trusted host to determine whether it has infected a real host. After host B infects m real hosts, it will pass the test and then will request any host in its peer list to download the main-force code.

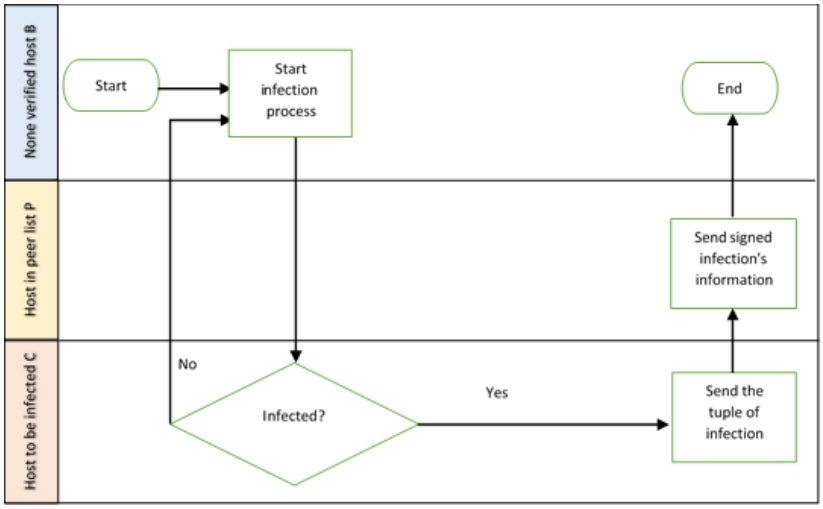

Figure 1: ATSRW method flow chart.

2.2. Avoiding Method (ATSRW-Aware)

The counter mechanism presented in [13] uses the IP addresses in the peer list. These addresses belong to infected hosts that have already joined the botnet. The mechanism works by remotely controlling at least one host from among those in the peer list. However, the goal of controlling the infected host is to pass the botnet authentication procedure by using a fake handshake process, and not monitoring the botnet.

Figure 2 shows the flowchart of the method. After the honeypot B gets infected by spearhead code, B will attempt to compromise at least one peer from its peer list. Suppose P (a member of B's peer list) controlled by the honeypot B. Then, honeypot B will search for M arbitrary hosts to infect by sending the spearhead code that contains B's peer list. The honeypot program will redirect all the infection traffic to another honeypot H. Honeypot H will spoof the IP addresses of the M arbitrary hosts and start a fake handshake process with P. Peer P will respond to the connection with the correct destination address (M arbitrary hosts). Because P is controlled by the honeypot, we can

ISSN: 2233-7857 IJFGCN

redirect the response to H and provide H with the correct TCP sequence and acknowledgement number. H can now complete the three-way TCP handshake and send the correct infection tuple to peer P to sign.

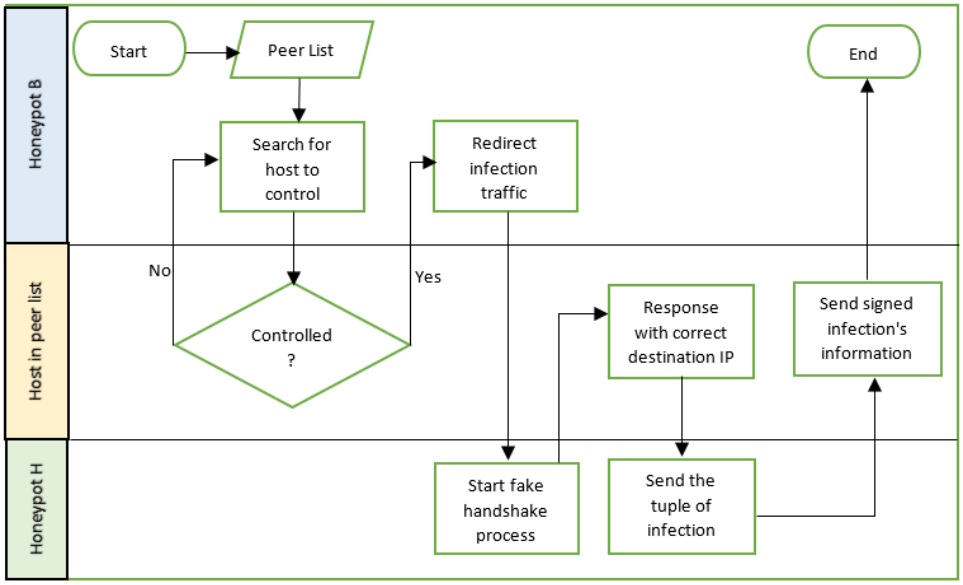

Figure 2: ATSRW- A ware method flow chart.

3. Proposed Method

The goal of the proposed method is to increase the chances of bypassing the authentication procedure used by the ATSRW method developed by Wang [10]. The proposed method is based on the idea of using some features of the botnet against itself. This concept has been proposed in several studies, such as [6], [7], and [8]. The main idea of the proposed method involves taking advantage of the M arbitrary IPs that were generated by the spearhead code. Although these M hosts are vulnerable, they are not yet infected by the botnet. Based on the security constraint, which was presented in [10], it is not allowed for honeypots to participate in real attacks that could cause damage to others. In our method, the honeypot will not harm the M hosts; instead, it will command them to pretend to be infected hosts. In addition, the honeypot will help break down the constructed botnet. The proposed method is illustrated in Figure 3. After honeypot B gets infected by the spearhead code, B will attempt to infect arbitrary hosts by sending spearhead that contains B's peer list. The honeypot program will redirect all connects to another honeypot H. H will attempt to compromise and control the arbitrary hosts whom B is trying to infect, knowing the bug used in the botnet. H will not infect the M hosts. Instead, it will command each of these hosts to contact the hosts in B's peer list using the TCP connection.

Thus, our mechanism needs:

Two honeypots: The first honeypot (B in Figure 3) should install the most popular operating system used by the end user. Steven in [14] has shown that Windows OS is the most frequently used OS. Therefore, the first honeypot will focus on vulnerable and not updated versions of Windows OS as an easy infection target for some botnets. The second one (H in Figure 3) may install any available OS.

A tool to redirect the traffic to another device.

A tool to help in accessing such a system by knowing its IP.

A simple code that commands the compromised host to send the infection tuple is also needed.

ISSN: 2233-7857 IJFGCN

This counter-mechanism will always succeed because the worm inside the honeypot will always scan for new IPs until it finds M vulnerable hosts to compromise. Thus, the host list is not limited as in the ATSRW-Aware method developed in [13]. In other words, the chance of success of this method is the same as that of the ATSRW.

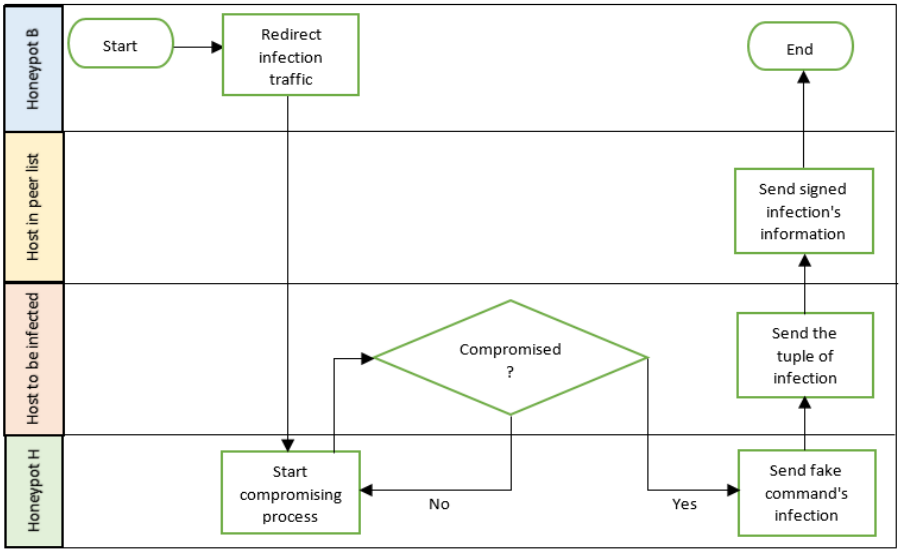

Figure 3: Proposed method flow chart.

4. Analytical Study

This section will measure the delay time of our method compared to the ATSRW and advanced two-stage reconnaissance worm aware method (ATSRW-Aware). We will follow the same analysis procedure presented in [13], which is based on the binomial probability distribution.

In ATSRW, the botnet adds the infected host after it has executed the main-force code. Thus, the maximum time delay that allows the infected hosts to join a botnet is [13]:

$$D = SpearT + M \times (SpearT + PeerLT) + AccessHT + M \times K \times TupleT + M \times K \times ReportT \quad (1)$$

Where, SpearT is the spearhead transmission time, PeerLT is the peer list transmission time, TupleT is the infection tuple transmission time, ReportT is the report transmission time which contains the signed infection tuple, M is the arbitrary hosts that should be infected, K is the number of available peers, and AccessHT is the time to compromise M hosts which equals:

$$AccessHT = (t + L/2)M\lambda/\alpha + LM(1-\lambda)/\alpha \quad (2)$$

Where, t is the delay to compromise an online host, L is the maximum waiting time for the host to be online, λ the probability of a host being available (online) before time ^{L}, and α is the probability of a host being both available and vulnerable.

The transmission time is:

$$Transmission\,time = PacketLength / R \quad (3)$$

ATSRW-Aware method allows infected honeypots to join a botnet after compromise some peers in the peer list [13]. The time delay is:

$$D1 = SpearT + M \times (SpearT + PeerLT) + Y \times AccessPT + M \times Y \times (TupleT + SpoofT) + M \times Y \times ReportT \quad (4)$$

ISSN: 2233-7857 IJFGCN

Where, SpoofT is the time of the spoofing process, Y is the number of compromised peers, and AccessPT is the time to compromise one peer.

Let the delay to compromise an online peer be t. The average time to find an available peer is L/2, the waiting delay for an offline peer is L, and the probability of a peer being vulnerable is αp.

The average delay required to compromise a peer in the peer list is:

$$AccessPT = (t + L/2) \times Y\lambda/\alpha p + L \times Y(1-\lambda)/\alpha p \quad (5)$$

The proposed method allows infected honeypots to join a botnet with a certain time delay, which is:

$$D2 = SpearT + M \times (SpearT + PeerLT) + AccessHT + M \times CommandT + M \times K \times TupleT + M \times K \times ReportT \quad (6)$$

Where, CommandT is the fake command transmission time.

The success of this method depends on taking control of the M arbitrary hosts that the botnet wants to infect. These hosts are vulnerable and contain all the holes that the botnet needs. Therefore, success depends on our capability to control them. We will suppose that as the ATSRW succeeds, the method will succeed in controlling the M hosts. The time needed to compromise the M hosts is the same as that required by the botnet. Therefore, the AccessHT is same as in equation (2).

5. Performance Results

The ATSRW method delay was simulated by the Monte Carlo method using JAVA programming language. The Monte Carlo method was used to determine the numerical value of the performance measure by using random generated inputs. The generator method selects a random number distributed uniformly over the interval]0,1[[15]. The Monte Carlo method is based on the "law of large numbers". This result is obtained if a large number of hosts/peers is generated from a specific range of vulnerabilities, V, and the delay of each is computed. The mean of these delays will approximate the mean value calculated by using the delay function on the V [15]. This section will show the results of the simulation using ZeroAccess parameters as one of the P2P botnet malwares shown in Table 1 [16]. The scenario will be repeated 10,000 times, and the average values are shown. After that, we will compute and compare the delay of our method with the delay in the authentication procedure of ATSRW and ATSRW-Aware.

Figure 4 shows the pseudocode of M compromising hosts in the ATSRW method. The delay simulated by searching sequentially in a network that contained 1000 hosts. Figure 5 shows the delay simulation of the ATSRW, as shown, the delay of this method decreases when the value of the vulnerable probability increases. The number of hosts added is shown in Figure 6. Figure 7 shows the pseudocode of compromising a number of peers in ATSRW-Aware method. As shown in Figure 8, the time delay (D1) decreases when the number of accessed peers (Y) decreases. The number of honeypots added is shown in Figure 9. The pseudocode of the M compromising hosts in our method was simulated as ATSRW. Figure 10 shows the delay simulation of our method to bypass ATSRW. The number of honeypots added is shown in Figure 11. The delay in the proposed method was in the range of the delay in ATSRW-Aware method, as shown in Figure 12. However, the proposed method gave us more time to compromise the M hosts.

Based on the results, we conclude that security defenders can use either the ATSRW-Aware method or our proposed method to bypass ATSRW. However, we recommend using the ATSRW-Aware method when the number of hosts that should be infected (M) is very high. We recommend using our proposed method when the number of online peers

ISSN: 2233-7857 IJFGCN

(K) is very small and difficult to compromise. If both conditions exist, we do not guarantee that the ATSRW procedure will pass without attracting the attention of the botmaster because of the high delay that exceeds the ATSRW. ATSRW-Aware method requires good raw-socket programming skills to implement the fake TCP connection, but our method requires a fake command, which can be easily written using any programming language. Nevertheless, the potential for unrealistic scenarios occurs in our method when the number of arbitrary hosts to be commanded is very high

Table 1: Botnet parameters

Parameters	Value
S	256 bots
R	72 Mbps
Spearhead length	52535 byte
Peer list length	65535 byte
Tuple length	600 bit
Report length	700 bit
Spoofing packets length	1296 bit
Fake command length	40960 byte
M	5 hosts
L	120 seconds
λ	0.30
t	100

```
For all nodes in the network do
    If selected host is online
        AccessHT += random number * L + t
        If selected host is compromised
            Number of compromised hosts incremented by
            one
    Else
        AccessHT += L
    If Number of compromised hosts equals M
        Break
End for
```

Figure 4: Pseudocode of ATSRW method.

ISSN: 2233-7857 IJFGCN

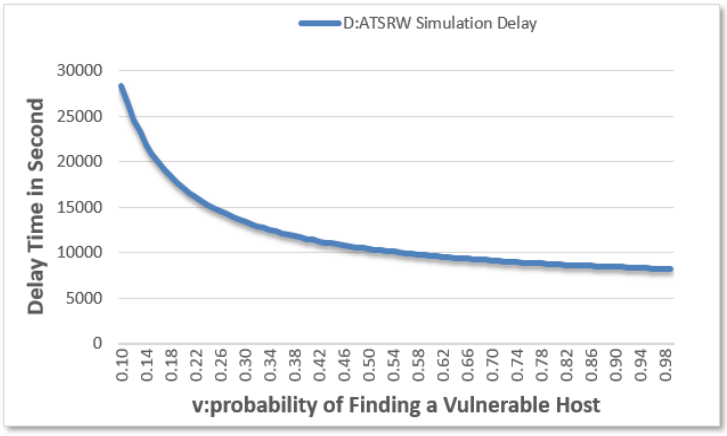

Figure 5: ATSRW delay.

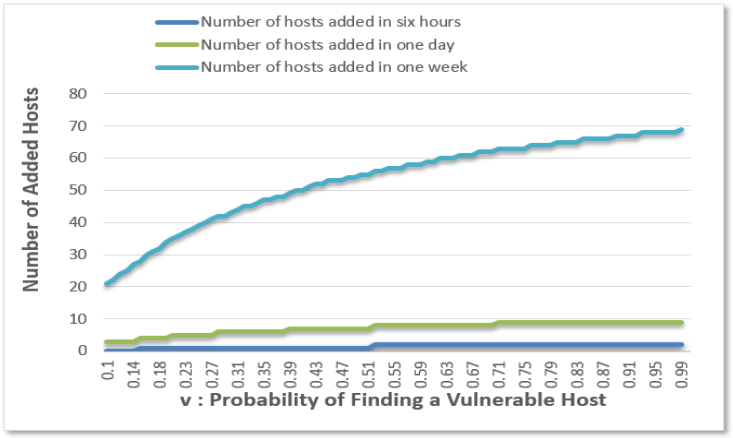

Figure 6: Number of added hosts using ATSRW

```
For all nodes in the peer list(S) do
    If selected peer is online
        AccessPT += random number * L + t
        If selected peer is compromised
            Number of compromised peers(Y) incremented by one
    Else
        AccessPT += L
    If Number of compromised peers(Y) equals the required
    number
        Break
End for
```

Figure 7: Pseudocode of ATSRW-Aware method.

ISSN: 2233-7857 IJFGCN

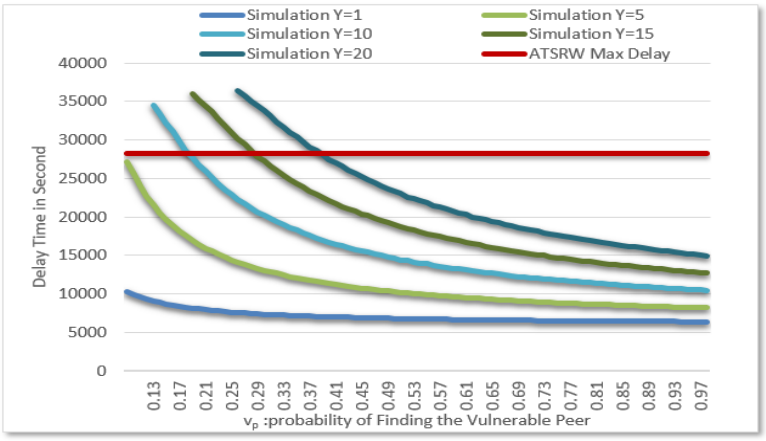

Figure 8: ATSRW-Aware delay

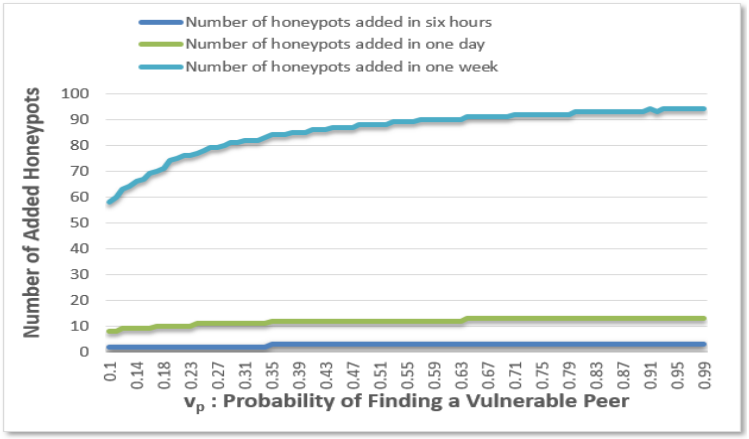

Figure 9: Number of added honeypots using ATSRW-Aware.

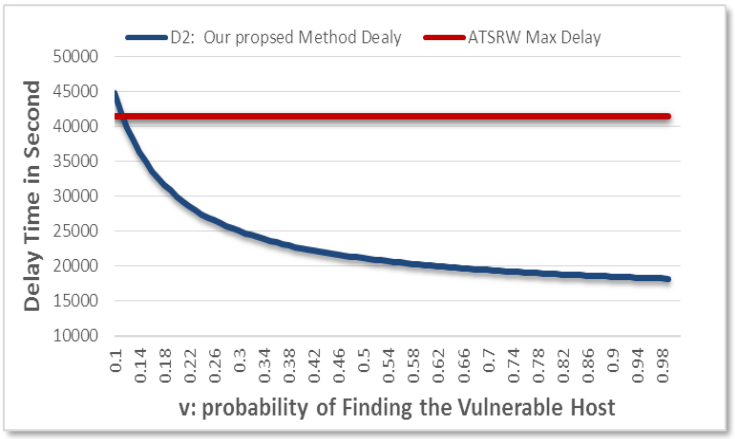

Figure 10: Proposed Method delay

ISSN: 2233-7857 IJFGCN

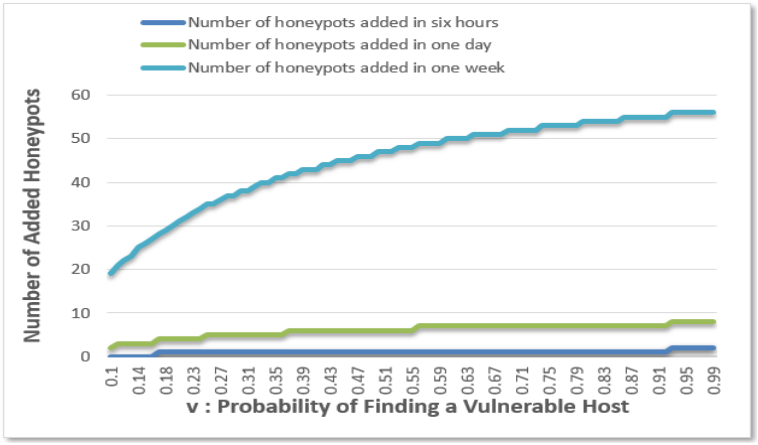

Figure 11: Number of added honeypots using the proposed method.

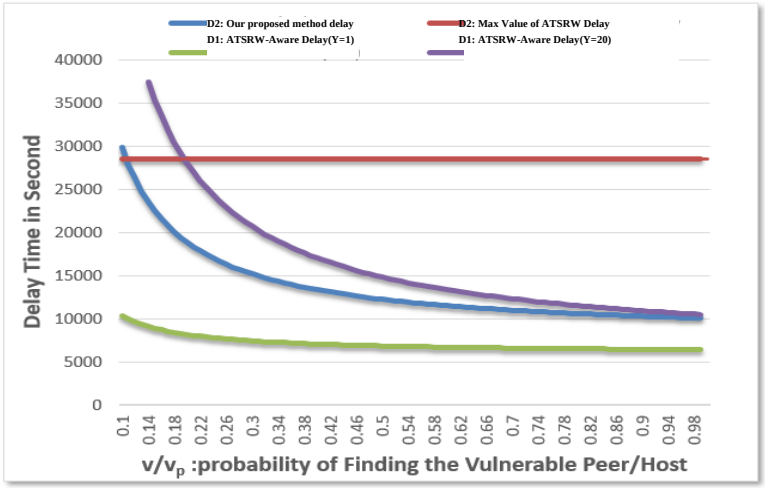

Figure 12: Comparison between methods.

6. Conclusion

In this research, we have developed a method to increase the means of adding honeypots to P2P botnets. These botnets have defence mechanisms that prevent honeypots from joining and monitoring them. The proposed method exploits the vulnerability used to spread the worm to create a fake infection on the hosts that are infected during the ATSRW authentication process. Using this, these hosts complete the authentication process just as a real infected host does. We have provided a simulation of the performance of that method in terms of the delay. We have also shown that the probability of success of this method is the same as ATSRW. We show that the delay of this method is smaller than the ATSRW delay and in the same range as the delay in the ATSRW-Aware method. In the future, studies are expected to focus on improving the methods to work well under both conditions: hard compromised peers and a large number of hosts that should be infected.

Acknowledgements

The authors thank the Deanship of Scientific Research and RSSU at King Saud University for their technical support.

References

[1] J. Kok and B. Kurz, "Analysis of the BotNet Ecosystem," CTTE, Berlin, 2011.
[2] M. Bailey, E. Cooke, F. Jahanian, Y. Xu and M. Karir, "A survey of Botnet Technology and Defenses," IEEE, Washington, 2009.
[3] J. Massi, S. Panda, G. Rajappa, S. Senth and S. Revankar, "Botnet Detection and Mitigation," 2010. [Online]. Available: http://csis.pace.edu/~ctappert/srd2010/c4.pdf. [Accessed March 2019].
[4] B. Asalm, C. Zou and P. Wang, "Peer-to-Peer Botnets: The Next Generation of Botnet Attacks," Pennsylvania State University, US, 2010.
[5] p. Wang, L. Wu, B. Aslam and C. Zou, "A Systematic Study on Peer-to-Peer Botnets," IEEE, CS, 2009.
[6] F. Castaneda, E. Sezer and J. Xu, "WORM vs. WORM: preliminary study of an active counter-attack mechanism," ACM, USA, 2004.
[7] F. Leder, T. Werner and P. Martini, "Proactive botnet countermeasures: an offensive approach," The Virtual Battlefield: Perspectives on Cyber Warfare, pp. 211-225, 2009 .
[8] C. Xiang, Y. Lihua, J. Shuyuan, H. Zhiyu and L. Shuhao, "Botnet spoofing: fighting botnet with itself," in Security and Communication Networks 8, Wiley, 2013, pp. 80-89.
[9] A. Borkar, A. Salunke, A. Barabde and N. Karlekar, "Honeypot: A Survey of Technologies, Tools and Deployment," ICWET, India, 2011.
[10] P. Wang, L. Wu, R. Cunningham and C. Zou, "Honeypot detection in advanced botnet attacks," International Journal of Information and Computer Security, vol. 4, no. 1, pp. 30-51,
[11] C. Zou and R. Cunningham, "Honeypot-Aware Advanced Botnet Construction and Maintenance," IEEE, PA, 2006.
[12] P. Wang, "Study of the Honeypot-Aware Peer-to-Peer Botnet and Its Feasibility," University of Central Florida, U.S, 2010.
[13] M. Al-Hakbani and M. Dahshan, "Avoiding Honeypot Detection in Peer-to-Peer Botnets," IEEE, India, 2015.
[14] V. Steven, "Today's Most Popular Operating Systems," 2017. [Online]. Available: https://www.zdnet.com/article/todays-most-popular-operating-systems/. [Accessed March 2019].
[15] M. Guizani, A. Rayes, B. Khan and A. Al-Fuqaha, "Monte Carlo Simulation," in Network Modeling and Simulation: A Practical Perspective, John Wiley & Sons, 2010, pp. 69-95.
[16] M. Mimoso, "Untouched P2P communication infrastructure keeps ZeroAccess up and running," 2013. [Online]. Available: https://threatpost.com/untouched-p2p-communication-infrastructure-keeps-zeroaccess-up-and-running/103133. [Accessed March 2019].

ISSN: 2233-7857 IJFGCN