ISSN: 1803-3814 (Printed), 2571-3701 (Online)
https://doi.org/10.13164/mendel.2020.1.007

X-SWARM: THE UPCOMING SWARM WORM

Thanh Cong Truong[1,2], Quoc Bao Diep[1], Ivan Zelinka[1,✉], Tran Trong Dao[3]

[1]VSB-Technical University of Ostrava, Department of Computer Science, Faculty of Electrical Engineering and Computer Science, Ostrava, Czech Republic
[2]University of Finance - Marketing, Ho Chi Minh, Vietnam
[3]Division of MERLIN, Faculty of Electrical and Electronics Engineering, Ton Duc Thang University, Ho Chi Minh, Vietnam
cong.thanh.truong.st@vsb.cz, ttcong@ufm.edu.vn, diepquocbao@gmail.com, ivan.zelinka@vsb.cz✉, trantrongdao@tdtu.edu.vn

Abstract

With the rapid growth of technology in the digital landscape, cybercriminals attempt to utilize new and sophisticated techniques to autonomous and increase the speed and scale of their attacks. Meanwhile, the Dark Web infrastructures such as Tor, plays a crucial role in the criminal underground, especially for malware developers' communities. It is logical to expect that the malicious actors would utilize the combination of these techniques in shortcoming time. To better understand the upcoming threat, in this manuscript, we investigate the design and mitigation of such malware. Accordingly, we introduce X-sWarm, which will be the next generation of resilient, stealthy malware that leverages the intelligent technique and the darknet infrastructures. Furthermore, we show that with the self-healing network mechanism, X-sWarm can achieve a low diameter and a low degree and be robust to partitioning under node removal. More importantly, we suggest the mitigation technique that neutralizes the nodes of the proposed worm.

Keywords: *Swarm intelligence, malware, Tor, cybersecurity, self-healing network.*

Received: 07 June 2020
Accepted: 02 August 2020
Published: 24 August 2020

1 Introduction

Recent years have witnessed a dramatic growth in utilizing computational intelligence techniques for various domains. Based on developments of evolving trends of cyber-threat, it is reasonable to predict that cybercriminals will begin to integrate malware with artificial intelligence in general, swarm intelligence technology in particular, to create more effective attacks, as stated in the literature [10, 11, 12, 13]. Generally, artificial swarm malware can share the collected information, speed up the process of trial and error, and leverage the specialized members of the swarms in the specific environment [8, 17, 14].

Alongside that, an emergency trend needs to be concerned is that the abuse of anonymity networks like Tor to evade detection and anonymize the location of the command and control (C&C) servers. With the deployment of Tor, a device can build a web-based hidden service (HS) for accepting connections without revealing their physical location.

A natural question which arises is what happens if the two mention technique is combined. To seek the answer to this question, we design a prototype called X-sWarm, which combines the above techniques to conduct analysis and understand their potential and limitations. From this result, we suggest developing the mitigation techniques for this kind of upcoming threats.

In previous studies [8, 17, 14], we explored the ability to integrate the AI in general and swarm intelligence in particular into the malware. In this manuscript, we assess the threat of malware with swarm behaviour that relies on Tor infrastructure. Accordingly, we present a design of the first generation of an X-sWarm, in which the communication channel is established through hidden services. We also propose a graph maintenance algorithm with high resiliency and repair in the event of a take-down. Furthermore, we also suggest the countermeasure technique based on the same stealthy features of the X-sWarm. Our main contributions are summarized as follows:

- We propose a novel reference design of a new type of malware with swarm characteristics, whose command, communication, and management are fully anonymized by leveraging the Tor privacy infrastructure.
- We define a communication topology with self-repair mechanisms that enhance the resiliency and performance of the network.
- We discuss the possible countermeasures to mitigate similar threats in the future.

The rest of this manuscript proceeds as follows. Section 2 introduces basic information that is involved with the research. Section 3 describes the methodology to design the worm. Section 4 present the evaluation process of the proposed concept. Section 5 suggests the ideas for countering the upcoming threat. And finally section 6 concludes the paper.

2 Background

In this section, we present some of the basic concepts that will be useful to better explain the proposed method by presenting significant objects.

2.1 Worm

Peter Szor, in his research [9] described the worm was a subclass of computer viruses but primarily propagated on networks. The main difference between a computer virus and worm is the propagation mechanism. While the virus spreads by infecting files on computer or network, worms usually propagate as independent programs. A copy of a worm will be called a worm instance to avoid ambiguity. Furthermore, worms can exploit the vulnerabilities of the remote system and compromise these systems without the assisting of the user. Today, many worms act a carrier for other malware, such as trojan horses and bots.

A typical worm contains the following components: target locator, infection propagator, payload routines self-tracker, and life-cycle manager. In this structure, two key components are the target locator and infection propagator; the other components are non-essential and vary according to the worm.

2.2 Tor and hidden services

Tor, which name derived from the acronym of the project name "The Onion Router", is a distributed low-latency anonymity-network [3]. It aims to help user protecting their privacy, circumvent censorship, as well as keep the user's confidential communication un-monitor [6]. What is more, the users capable of concealing their activities and location by using Tor.

Users establish anonymous communications by forwarding their traffic through other Onion Router (OR). The client negotiates with each relay in the circuit a separate set of encryption keys in order to enhance the privacy on the circuit. The client negotiates with each relay in the symmetric circuit locks to enhance the privacy on the circuit. Next, clients transmit data using an encrypted channel, using previously negotiated keys. The data are delivered from the relay to relay until it reaches the destination. In addition to providing anonymous communications, Tor also allows publishing services inside the network anonymously, which called hidden services.

3 Methodology

In this section, the author describes the methodology to design the prototype. The X-sWarm consists of the following components: target selection, infection propagator, communicator, payload. These components integrated to compromise a machine.

3.1 Target selection

This component is responsible for discovering new targets in order to spread the worm through the network. This is crucial to the success of the worm. There are several methods to identify the targets, such as through email addresses, network neighbourhood, or Random generation of target IP addresses.

For demonstration purpose, the target engine is implemented with a simple method. First, the worm discovers the IP address of the host. Next, it uses the Class-C boundary of that IP address and commences an Internet Control Message Protocol (ICMP) scan from A.B.C.0 through A.B.C.255. If a host responds to the ICMP echo request, the worm adds the host to the target list. Contemporary, the worm attempt to establish a Server Message Block (SMB) NULL Session when traversing through the IP address range.

3.2 Infection propagator

This part contains the strategy which is used by the worm to propagate itself to a new bud. Generally, there are some typical approaches to propagate the worm: Through security vulnerabilities, email, shared folder, or instant messaging.

In the scope of this research, we examine the null session technique to spread the worm. A null session is the unauthenticated sessions of the SMB protocol enables anonymous access to hidden administrative shares on a system. Consequently, the user can enumerate information about the system and environment when connecting to the share through a null session.

3.3 The propagation process

The propagation process starts by examining the local network address space, attempting to spread to as many machines as it can locally. After attempting to exploit all computers on the local network, it tries to intrude random external IP addresses.

The first thing when the worm tries to compromise a machine is by establishing a null session. This step provides the information on whether the machine it is attacking supports the CIFS protocol and is likely to be a Windows machine. Next, the worm uses the SMB protocol to enumerate the list of account names on the remote machine. It also establishes some basic properties about the user for guessing the password process. After that, it makes an SMB connection with the target computers, attempting to access the IPC$ connection. If the worm successfully connects to the IPC$ share, it copies itself over to the remote machine. After the copy process, the worm uses remotely schedule a job to run itself on the target machine.

3.4 Communicator

This component is responsible for communication in the swarm. In fact, we design a virtual network between infected computers to establish communication

and control operations. Each worm has a list of other known, running copies of the worm and capable of creating encrypted communication channels to spread information. This virtual network is built based on swarm intelligence principle. Such a network could be utilized to pass information rapidly to all running executables, lead to decentralize the C&C communication, and preventing the communication channel from being disrupted by others, make the worm hard to track. The two techniques are utilized for transmitting information are TCP and through Tor network.

In our design, the prototype would form a peer-to-peer (P2P), self-healing network that maintains a low degree and a low diameter with other instances to relay messages. In the following, we present an abstract graph representation of a worm communication topology, which is capable of self-repairing and dynamic distributed. This communication topology is simple, stealthy and resilient, which formed over a privacy infrastructure such as Tor.

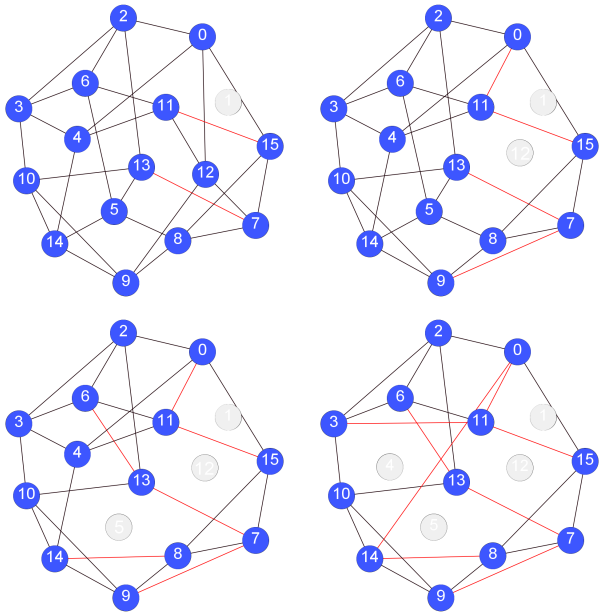

Figure 1: Node delection and the self-healing process

3.4.1 Graph structure

We propose to use the concept of Neighbour-of-Neighbour (NoN) [7] to construct the abstract graph. In the literature [7], the authors examined the neighbour of neighbours for making better routing decisions. In the scope of this study, we investigate the concept of NoN to create a self-healing network. Note that in this study, we use vertex and node with equivalent meaning. We define our graph as below:

Definition. Consider a graph G having n vertices (V) and m edges (E), where each vertex $v_i \in V, 0 \leq v < n$, is linked which a number of vertices. The set of neighbour verices of v_i is denoted as $N(v_i)$. Furthermore, the vertex v_i has the information of vertices that are linked to $N(v_i)$. In other words, each vertex knows the information of its neighbour of neighbour.

3.4.2 Network self-healing mechanism

Based on the neighbour of the neighbour graph, we propose a network self-healing mechanism to form a new connection and substitute the relay function of the removed node. Thus, the connectivity of the network is maintained. The specific process of the repairing is: Assuming a vertex v_i is deleted, the neighbours of v_i react to this deletion by adding some set of edges amongst themselves. These edges can only be between nodes which were previously neighbours of v_i. This is to ensure the locality information in the underlying network is maintained after inserted the edges.

One aspect to take into account is that the insertion of the new edge may lead to the growth in the connectivity degree of each vertex, denoted by $\delta(v)$. Indeed, the degree of some vertices may rise significantly after repeated deletion. Nevertheless, increasing the degree of such vertices is undesirable for the worm's resilience and clandestine operation. Hence, we propose to keep the degree of the vertices in the range $[\delta(Min), \delta(Max)]$ when add a new edge.

Figure 1 depicts the self-healing process in a 4-regular graph with 16 nodes. The red lines indicate the newly established connection between the vertices. For example, when vertex 1 is deleted, its neighbors $N(1) = \{7, 11, 13, 15\}$ react to this deletion and traverse their neighbor list to check whether they are linked to each other, and then establish a new one. In this case, the following edges are created: (7, 13), and (11, 15). Similarly, when vertex 12 is removed together with its connected edge, the new edges are appeared: (7,9) and (0, 11).

3.5 Command and control communication

In our prototype, communication is entirely encrypted by using Tor and Secure Sockets Layer (SSL). Furthermore, the encryption keys are unique to each link. Additionally, there is no central server; instead, all requests are handled by peers within the network. Each worm member acts as a command server and a client. Consequently, this structure helps the worm more resilient against the defences method than the traditional centralized structure.

The communication inside the swarm relies on the peer list that contained in each worm. This list is fixed and has a limited size for each worm. Thus, when a worm is revealed, only a few numbers of worms in its peer list are exposed. To forward command, a worm could use its neighbours as targets and rely on these neighbours to continue passing on the command in the swarm worm.

Furthermore, the peer list based architecture can be utilized to implement strong encryption as suggested in [15]. Technically, each worm i generates its symmetric encryption key K_i. Assume the worm x has its peer

list, which is denoted by PL_x. This peer list would consist of not only the N .onion address but also the symmetric keys of its neighbours. Hence, the peer list on worm x is:

$$PL_x = \{(O_{i_1}, K_{i_1}), (O_{i_2}, K_{i_2}), ..., (O_{i_n}, K_{i_n})\} \quad (1)$$

where (O_{i_j}, K_{i_j}) are the *.onion* address and symmetric key used by the worm i_j. This encryption ensures that if a worm is captured, then just the keys in the captured worm's peer list are revealed. Hence, the encryption among the remaining worms will not be endangered.

3.6 Payload

A worm's payload is designed to perform specific actions on behalf of the worm's author on the victim system. In this prototype, the payload is to test the swarm worm functionality. Hence, no destructive payload was implemented, except spread to other machines.

3.7 X-sWarm swarm behaviour

Over the past few years, we have seen that traditional worm and botnet has a critical weakness that is the centralise C&C communication. Thus, cyber-threat actors attempt to discover a different method to overcome this disadvantage. One potential approach is to leverage the swarm intelligence (SI) to overcome the centralise weakness. With the latest advances of swarm technology, it is logical to expect that in the upcoming time swarm intelligence (SI) will be utilised to obtain this goal. Hence, to deal with this future threat, we need to have knowledge about this emergence trend so that we can design an efficient solution for countering the SI-based malware threats. For this reason, in this work, we propose a new swarm-based C&C behaviour in malware network. We aim to decentralise the infrastructure as well as autonomy the role of each member in such a network.

Accordingly, we suggest to design a P2P malware network that is able to share information – between malware nodes – and act on their own without a malware author issuing any commands. In this network, the nodes capable of communicating with each other and shared local intelligence. For example, the malware attempts to learn information about a potential victim, and when it discovers the victims, it will share this information for the rest of the swarm. Furthermore, each node can make autonomous decisions with minimal supervision, use the collective intelligence to solve problems. This network allows node executes commands without the central instruction, and recruit and train new members of the swarm. Consequently, as a swarm compromises the more devices, it will be able to grow exponentially and thereby enhance its ability to attack multiple targets simultaneously.

In order to the swarm operate autonomously, the communication-feedback mechanisms are required. We suggest that each peer in the P2P network supports bidirectional commands, enabling a peer within a network request and receive a response. Accordingly, each peer contains a set of commands that allow it to interact with other peers using custom-built P2P communication for performing multi-tasks routines. In the context of our work, the Tor protocol is utilised as a communication channel. Table 1 depicts the necessary commands for the communication process.

4 Evaluation

In this part, we conduct several experiments to prove the concept that is proposed in the previous section. More precise, we simulate to evaluate the resiliency and performance of the self-healing algorithm. The experiment involves the simulation process of the self-healing network resiliency and performance. For studying the robustness and the attack tolerance of networks, we conduct the procedure of removing a node from a network, where the node is chosen randomly, which mean the removal of a set of nodes happens with a certain probability.

4.1 Efficiency evaluation

The efficiency is a measured metric that needs to be concerned when study communication in the network. A worm may be evaluated by its communication efficiency, such as how long it would take to transmit messages, update binary code, or collect the host's information.

To investigate the effect on the network connectedness, we simulate and construct some generic models to analysis. Among various existing models for generating networks Erdos Renyi model [5] of the random networks and the Barabási Albert model [2] of the scale-free network are widely used. Hence, in the simulation process, we simulate the following models: two regular models ($N = 1000$, $k = 4$) in which one have the self-repairing mechanism, an Erdos Renyi model ($N = 1000$, $p = 0.05$) and an Barabási Albert model ($N = 1000$, $m = 5$).

Figure 2 illustrates the results for the vertex attack vulnerability measured by the average inverse shortest path length l^{-1}. As shown in Figure 2, the regular model decay exponentially after 20% of nodes is removed, while with the Erdos Renyi model, the rate is 30%. This can be explained from the finding that each node has approximately the same degree and thus contributes to the network by relatively the same amount. For the Barabási Albert model, the l^{-1} slightly decreases until 50% of node removal. This is of course, due to the large variation in the importance of the nodes, i.e., there exist significant vertices, hubs. These hubs act as a crucial role in network functionality. As long as the hubs are not eliminated, the connectivity of the network remains.

On the contrary, in the regular model applying our proposed self-healing, as the nodes are deleted and the number of nodes decreases, the l^{-1} of the graph also

Table 1: List of command establish the autonomous of swarm.

Type of command	Description
Peer list	These type of commands are utilised to maintain the peer list up to date
Update config	The update commands are leveraged to ensure that every member have the latest config
Report	These commands are responsible for reporting the potential target.
Data transfer	These commands allow to transmit data between peers

slightly rises accordingly, even when 90% of the node is deleted. This should be interpreted as the network functionality remains after removing a large of nodes. Furthermore, as the number of nodes decreases, the average of inverse shortest path length rises, which mean the dynamics of the network increases. Taken together, these results show that our algorithm helps maintain connectivity and even increases the efficiency of the network when deleting vertices.

4.2 Robustness evaluation

To evaluate the robustness, we examine how resilient our network is to failures in the network, such as members being eliminated. We utilise some metrics that are used in graph theory, such as the closeness centrality, degree centrality and a number of the connected component after nodes removal.

To examine our proposed algorithm's ability to repair the network, some experiments are deployed. We simulate the node removal process in the network of 1000 nodes, with up to 90% (900) node deletions. Four models are utilised including two k-regular (k=4), an Erdos Renyi model, and an Barabási Albert model. In these models, one k-regular model applies our proposed self-repairing algorithm while the others use a naive self-repairing algorithm (each node may add edges joining it to any other neighbour nodes as desired). The figure 3 illustrates the mean degree centrality and the mean closeness centrality when deleting nodes. From the figure, it can be seen that the model apply our algorithm to keep the mean centralities stable. This result may be explained by the fact that our method keeps the degrees of the nodes in the bound constraint during the repairing process. On the other hand, in figure 3, there is a clear trend of increasing the degree and closeness centrality in the models that utilise the naive self-healing algorithm. From the result, we can see that the naive self-healing approach cause high degree increase (which may lead to overload and eventual network breakdown) or increase in distances between nodes (which may lead to poor communication). Whereas, our proposed method keep the metrics stable, and even after 90% of node removal, the degree centrality and closeness slightly increase. Low degree centrality is desirable because it decreases the chances of detection and takes down.

In order to understand how node removal affects the network, we simulate the nodes deletion process of four types of model: a normal 4-regular, an Erdos Renyi, an Barabási Albert, and finally a 4-regular model with self-healing mechanism. Figure 4 depicts the simulation result when removing nodes of two network of size 1000 (a) and 10000 (b), respectively. From the data in Figure 4, it is apparent that the self-repairing model remains connected even when a large portion (80%) of the nodes are deleted, compared to other types of the model (with no self-repairing mechanism). Note that, in a normal model after 30% node deletion, the number of partitions rise sharply. The similar phenomenon happens in Erdos Renyi and Barabási Albert model when removing 50% of nodes. From this result, we notice that the normal 4-regular model is the most vulnerable to random node removal whereas Erdos Renyi and Barabási Albert models have better tolerance with node deletion. On the contrary, the model with the self-healing mechanism is the most resilient network. It can, therefore, be assumed that our self-healing algorithm makes the k-regular network more resilient and robustness.

5 Countermeasures

In this section, we discuss some different mitigation strategies to counter against the X-sWarm. Mitigation and detection can take place at different levels, such as host level or network level.

5.1 Countermeasures for host level

Tor services frequently listen to several specific ports: ports 80, 443, 9001 and 9030, the default ports of the Tor protocol while it is running on an infected device. The communication is easily blocked by filtering network traffic if there is no application based on the Tor protocol on the infected device since X-sWarm relies on Tor for communicating. Generally speaking, there are two feasible ways to do this. The first approach involves with control the traffic outbound to the Internet by the ports being used, such as block outbound traffic to specific ports, or limit the permitted outbound traffic to certain ports. The latter approach concerns with using network inspection techniques to try and determine which is legitimate traffic and which is malicious traffic. Nevertheless, blocking some commonly used ports may affect the user experience. Therefore, traffic filtering can be a temporary countermeasure.

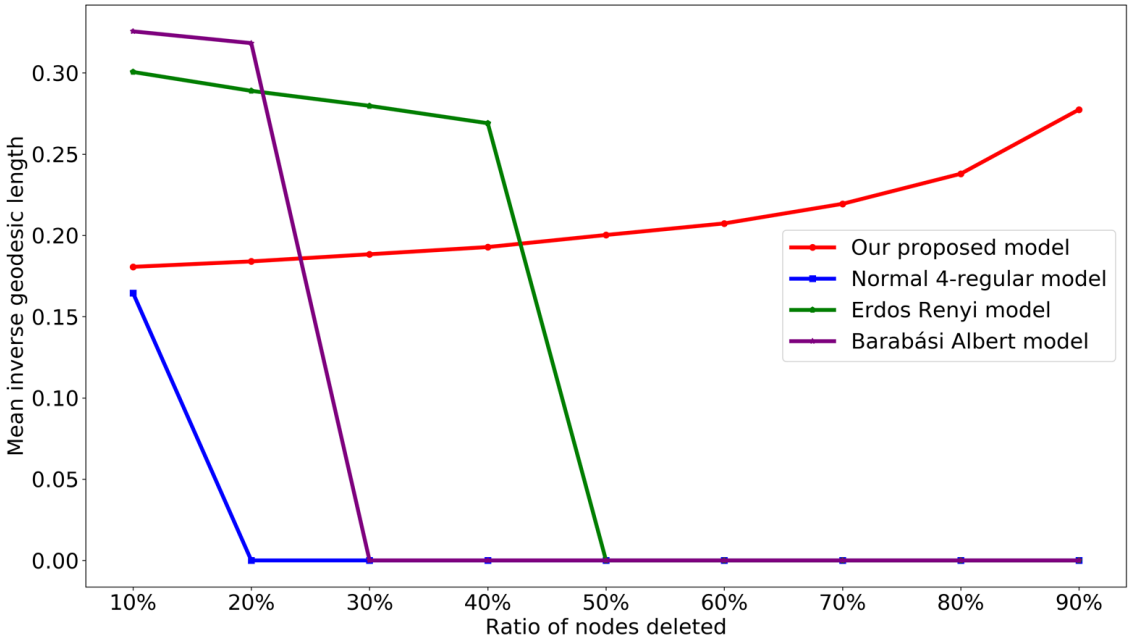

Figure 2: Mean inverse shortest path when removing nodes

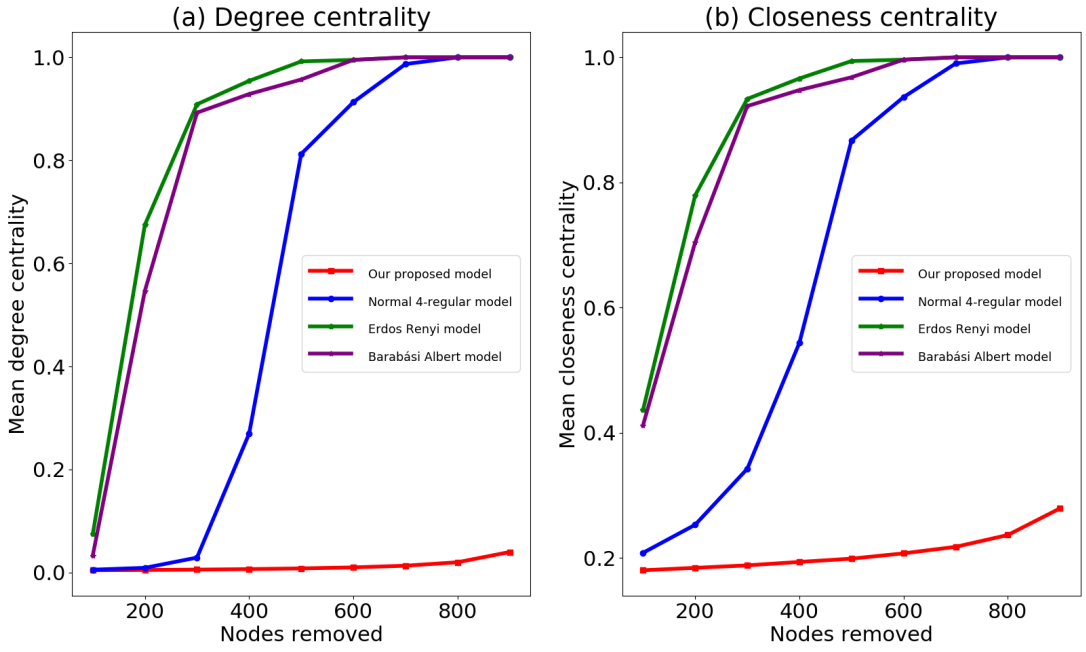

Figure 3: Mean centrality of the network when removing node

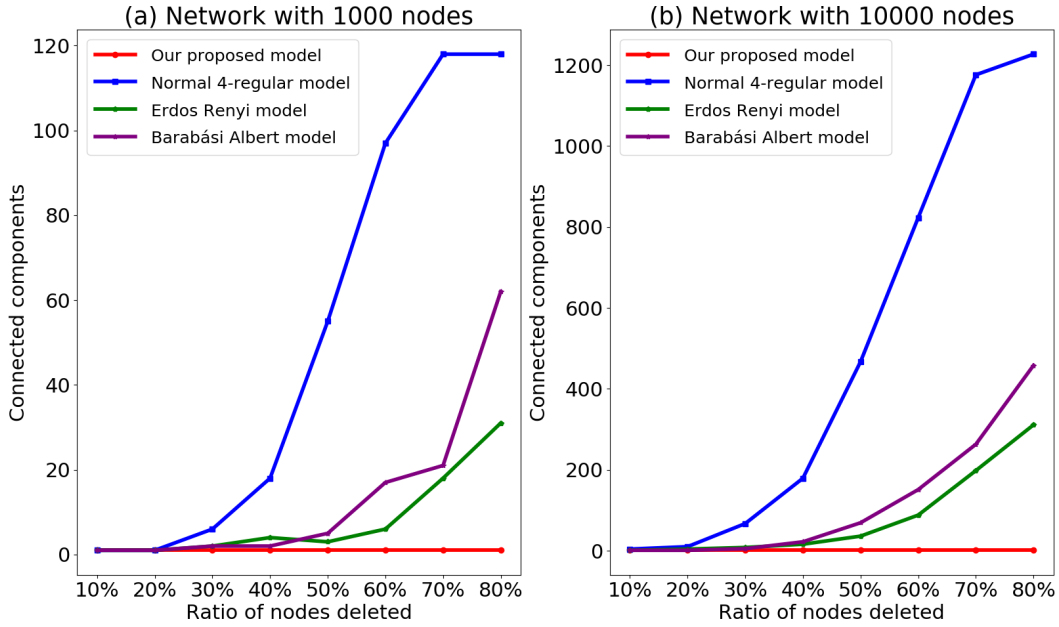

Figure 4: The number of connected components after removing a fraction of nodes

5.2 Network level countermeasure idea

"Sybil attack" is referred as a small number of entities forge multiple peer identities so as to compromise the peer-to-peer distributed systems [4]. In many P2P networks, the peers are feasible to join the network without authentication or validation of their identities. As a consequent, these P2P networks are vulnerable to Sybil attack.

In our work, we leverage the Sybil attack to form a countermeasure idea for the X-sWarm. To attack and break network down, the peer's onion addresses need to be obtained. This can be done either by detecting and reverse engineering an already infected host or by using a set of honeypots. After identifying the peer address, we run many hidden services, disclosing a subset of these as neighbours to each peer we encounter, so gradually over time our clone nodes dominate the neighbourhood of each peer and contain it.

6 Conclusion

In this manuscript, we present X-sWarm, a novel design of malware with swarm characteristics, in which communication utilizes the Tor network. This design has shown that the combination between SI and Tor network producing a robust and stealthy malware that has the ability to evade detection, measurement, scale estimation and observation. Additionally, X-sWarm relies on a resilient self-healing network formation that is simple to implement, yet robust to partitioning, even up to 90% node removal the network is feasible self-recover. The results demonstrate the feasibility, stealthiness, and robustness of this new type of malware. More importantly, we suggest the countermeasure approach as the host and network level for this upcoming threat. These findings contribute in several ways to our understanding of X-sWarm and provide a basis for further research.

This research has also opened research directions. One potential research area is applying new malware detection techniques [16, 1] based on the whole swarm activities to identify the malware. On the other hand, based on the X-sWarm idea, we could develop autonomous anti-malware technology in complex and large systems.

Acknowledgement: The following grants are acknowledged for the financial support provided for this research: Grant of SGS No. SP2020/78, VSB Technical University of Ostrava.

References

[1] Amer, E., and Zelinka, I. A dynamic windows malware detection and prediction method based on contextual understanding of api call sequence. *Computers & Security 92* (2020), 101760.

[2] Barabási, A.-L., and Albert, R. Emergence of scaling in random networks. *Science 286*, 5439 (1999), 509–512.

[3] Dingledine, R., Mathewson, N., and Syverson, P. Tor: The second-generation onion router. Tech. rep., Naval Research Lab Washington DC, 2004.

[4] Douceur, J. R. The sybil attack. In *International workshop on peer-to-peer systems* (2002), Springer, pp. 251–260.

[5] Erdős, P., and Rényi, A. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci 5*, 1 (1960), 17–60.

[6] Goldschlag, D., Reed, M., and Syverson, P. Onion routing. *Communications of the ACM 42*, 2 (1999), 39–41.

[7] Manku, G. S., Naor, M., and Wieder, U. Know thy neighbor's neighbor: The power of lookahead in randomized p2p networks. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 2004), STOC '04, Association for Computing Machinery, pp. 54—63.

[8] Sikora, L., and Zelinka, I. *Swarm Virus, Evolution, Behavior and Networking.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2018, pp. 213–239.

[9] Szor, P. *The Art of Computer Virus Research and Defense.* Pearson Education, 2005.

[10] Thanh Cong, T., and Zelinka, I. A survey on artificial intelligence in malware as next-generation threats. *MENDEL 25*, 2 (Dec. 2019), 27–34.

[11] Truong, T. C., Diep, Q. B., and Zelinka, I. Artificial intelligence in the cyber domain: Offense and defense. *Symmetry 12*, 3 (2020).

[12] Truong, T. C., Huynh, T.-P., and Zelinka, I. Applications of swarm intelligence algorithms countering the cyber threats. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion* (New York, NY, USA, 2020), GECCO '20, Association for Computing Machinery, p. 1476–1485.

[13] Truong, T. C., Zelinka, I., Plucar, J., Čandík, M., and Šulc, V. Artificial intelligence and cybersecurity: Past, presence, and future. In *Artificial Intelligence and Evolutionary Computations in Engineering Systems* (Singapore, 2020), S. S. Dash, C. Lakshmi, S. Das, and B. K. Panigrahi, Eds., Springer Singapore, pp. 351–363.

[14] Truong, T. C., Zelinka, I., and Senkerik, R. Neural swarm virus. In *Swarm, Evolutionary, and Memetic Computing and Fuzzy and Neural Computing* (Cham, 2020), A. Zamuda, S. Das, P. N. Suganthan, and B. K. Panigrahi, Eds., Springer International Publishing, pp. 122–134.

[15] Wang, P., Sparks, S., and Zou, C. C. An advanced hybrid peer-to-peer botnet. *IEEE Transactions on Dependable and Secure Computing 7*, 2 (2010), 113–127.

[16] Zelinka, I., and Amer, E. An ensemble-based malware detection model using minimum feature set. *MENDEL 25*, 2 (Dec. 2019), 1–10.

[17] Zelinka, I., Das, S., Sikora, L., and Šenkeřík, R. Swarm virus-next-generation virus and antivirus paradigm? *Swarm and Evolutionary Computation 43* (2018), 207–224.