ISSN: 1803-3814 (Printed), 2571-3701 (Online)
https://doi.org/10.13164/mendel.2021.1.018

Intelligent Malware – Trends and Possibilities

Jan Plucar, **Jiří Frank**, **Daniel Walter**, **Ivan Zelinka**

Faculty of Electrical Engineering and Computer Science VŠB-TU, Department of Computer Science, Czech Republic
jan.plucar@vsb.cz,jirifrankit@gmail.com, daniel.walter.st@vsb.cz,ivan.zelinka@vsb.cz

Abstract
In recent months and years, with more and more computers and computer systems becoming the target of cyberattacks. These attacks are gaining strength and the sophistication of the approach in terms of how to attack. Attackers and Defenders are increasingly using artificial intelligence methods to maximize the success of their actions. For a successful defence, we must be able to anticipate future threats that may come. For these reasons, our research group is engaged in creating experimental software with artificial intelligence to test the possibilities and capabilities of such malware in the event of its deployment. This software has not only malware capabilities but also antimalware and can be used on both sides. This article introduces the reader to the main principles of our design, which can serve as a future platform for cyber defence systems.

Keywords: *malware, artificial intelligence, swarm, artificial neural network.*

Received: 02 April 2021
Accepted: 27 May 2021
Published: 21 June 2021

1 Introduction

Today, technology is spreading to all areas of life. Entities connected to the Internet are no longer just computers and mobile devices, but also other smart devices and sensors. The current epidemic of Covid has also brought with it the need to work or attend school lectures from home. This brings new challenges. Millions of companies had to hand out personal computers to their employees in a short time, expanding the virtual perimeter of their workspace. Of course, the malware creators also took advantage of the situation and focused on this new extended cyberspace. Many reports from well-known security companies point to this fact.

In such a rapidly evolving environment, it is necessary to actively and quickly respond to new threats. The technology that could help us in this area is artificial intelligence (AI). In recent years, researchers have actively exploited the idea of using AI to detect malware [15]. Authors in [12] proposed malware detection model MalConv that was based on analysis of raw byte sequences of malicious executables. The paper discusses initial work and presents future challenges. MalConv model was further analyzed and expanded in [4], where authors tried to understand how the system learns to discriminate between malicious and benign executables using raw bytes. The main contribution of the paper includes gradient analysis at various stages of the trained network to see how the system assigns weights to different portions of the executable, analysis of the filter weights and their activations for different files.

At a higher level, malware can be analyzed from a static or dynamic perspective. Static analysis commonly uses Portable Executable (PE) metadata. This is a case of research in [7]. The authors presented a neural network for the detection and classification of malware based on information from static analysis. The neural network consists of convolutional and feedforward layers and uses metadata such as imported functions and series of opcodes to separate malicious executables from benign programs.

Other works are devoted to the combination of dynamic analysis and AI. For example, authors in [8] constructed deep neural networks to improve modelling and classification of system call sequences. System calls sequences were also used in [1] where authors used AI in combination with clustering methods to differentiate between malicious and benign software samples.

Naturally, artificial intelligence methods use not only defence systems but also malware itself. One example is shown by authors in [2], who used artificial intelligence methods to modify PE headers to avoid detection by static analysis.

Another example of malware that uses artificial intelligence is given in [17]. This article lays the foundations for a new type of malware, namely swarm-based malware [16]. We follow the basics of this article in this paper, where we will show the use of artificial intelligence in an illustrative example.

2 Motivation

The motivation for this research is the fact that the intensity of cyberattacks is increasing, and they are becoming more and more sophisticated. Just look at the recent past when various types of cyberattacks took place around the world. Just remember power outages in Argentina and Ukraine [5] or the recent attack on the US Colonial pipeline oil supply network. At present, the attacks focus not only on industrial structures but also on medical facilities and military [6] and state facilities. The goal of these attacks is to obtain information or, by ransomware, to obtain money. Therefore, it is clear that there is a growing need for new types of protection tools that will use non-traditional algorithms and approaches to prevent future threats that will also rely on artificial intelligence [17], [14], [11]. That is why the software described here is being developed, the ultimate goal of which will be to protect complex cybernetic systems [10]. Be one step ahead of the attackers.

3 Experiment design

This work aims to describe the possibilities of creating a covert communication channel in swarm malware. The main principles of swarm malware have been described in [17]. As part of the experiment, we will verify the process of creating conditions for communication between individual members in a local network which leads to the establishment of the communication between swarm members in a covert channel using Tor technology. Dynamic malware analysis has proven to be a powerful tool. As a complement to dynamic analysis, network analysis is commonly performed. The most common example of malware that now communicates over a network operates within a Bot network and usually uses DNS, HTTP, or IRC protocols. Members of the botnet commonly ask for instructions from the command and control node (CC), which they then execute. This phenomenon is usually repetitive and detectable by network analysis [13], [3].

Swarm-type malware does not support query for instructions from the CC node. Infected individuals in the swarm operate independently of the CC node during periods of inactivity and do not need to cooperate with the CC node because there is none.

In the spreading phase, future swarm members download the Tor and create an own onion domain, which they distribute to other active swarm members in the form of a random broadcast message in a local network, see Fig.1. Each swarm member encrypts the onion domain to the disk. Swarm members will collect as many onion domains of the known members as they can.

When the payload execution is necessary or any other cooperation across swarm, all members of the swarm can communicate with each other. Communication will take place unnoticed within the Tor network via onion domains. As part of our experiment, we do perform only simple payload task and focus mainly on communication between swarm members. Our swarm members are sending testing messages with the timestamps to measure the latency of the swarm network.

Ideally, members can communicate with each other. In case that one of the members is not able to communicate with another specific member, i.e. the specific member is blacklisted by a local network firewall, it is possible to inquire through an intermediary, i.e. another member on the network, to force blacklisted member to generate a new onion domain. Because of this, the whole swarm is more resistant to intervention such as the blacklisting of specific addresses. Another important difference that was already mentioned is that there is no CC node that can be blacklisted.

Our experiment is designed as follow. Each testing machine is a virtual instance. In the beginning, all machines are in the same local network. If any of them become infected, they inform the rest of the swarm and start to obtain onion domains. Since each communication is Peer-To-Peer, even if one host is disconnected or terminated, the network will continue to function as intended.

Suppose the infected machines are all part of the same internal network. All members of the network immediately download Tor, and their onion domains are exchanged. Exchange of onion domains continues only in one local network with the usage of broadcast messages. After that, all participants can communicate via the Dark web network and establish Peer-To-Peer communication. The next step in the experiment was to move some members from the first local network to a different local network. Even if swarm member is disconnected from their local network and switched to another, it should be able to communicate without any problems. One requirement is that there must be access to the Internet to make communication possible.

3.1 Used technology

Swarm malware was mostly programmed in C# with some tasks being written in PowerShell scripting language.

C# is very popular and offers fast development and expansion. However, this malware was created as a fileless version. Malware is saved to registry as an encoded string and simple launcher load this encoded string into memory and run the code under random process. The conclusion of this is running malware, which is hidden under different processes.

Last but not least was used Tor for communication. This protocol provides the advantage of establishing a connection to whatever location in the world. Communication between nodes is based on signals from ANN, and only members in ANN can understand what each signal determines. Some signals can inform members about changed weights, but some of them can give a command to run the payload. The payload for this malware can be anything. In this malware is existing ANN, which will decide which payload will be used based on network, infected hosts.

3.2 Malware structure

One particular malware sample, a member of the swarm, consists of several modules, see Fig.2. The structure of the virus is as follows. It contains the program module, which is responsible for basic behaviour. Another key part of the virus is a set of several modules that are in charge of neural network use and its management, which is still in the development phase. The

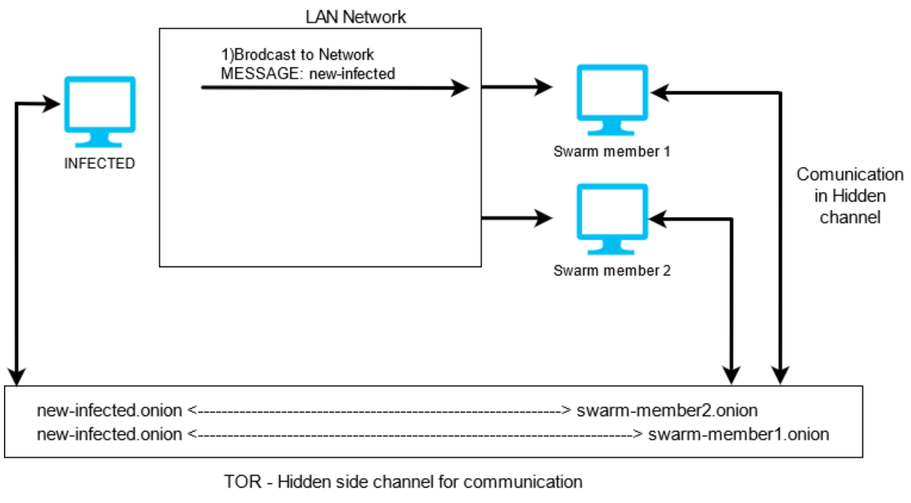
LAN Network
1)Brodcast to Network
MESSAGE: new-infected
INFECTED
Swarm member 1
Swarm member 2
Comunication in Hidden channel
new-infected.onion <--> swarm-member2.onion
new-infected.onion <---> swarm-member1.onion
TOR - Hidden side channel for communication

Figure 1: Swam member's communication

most important module of virus for this article is the module covering the functionality of the Tor network, which is used for communication.

The program module is in charge of the basic behaviour of itself. It contains threads that help him do several key things. Spreading, sending broadcast messages to other swarm members on the local network and retrieves sent data from other members on the local network. It also contains threads for communication via the Tor network. Other threads are dedicated to handling neural network.

The spreading module is not ready to use. We need to implement any already existing worm or dropper to test the whole concept, but it is turned off and not tested for current experiments.

Regarding neural networks, the two types of approaches are experimented with. The first approach looks like this. Each member of the swarm, i.e. a self-functioning unit, carries its own learned neural network, which can be used to propagate control signals across the swarm or encrypt and decrypt communication between individuals members of the swarm.

The second approach is different in the way of accessing the neural network itself. There is only one neural network that is redistributed across the whole swarm. Each member is caring one neuron or bigger parts of the neural network. In the second approach, the neural network is driven in a decentralized way. There is no one specific command point. The neural network can be redistributed across the swarm if necessary or healed if the neural network is broken. Decentralized redistribution and healing are possible because each swarm member carries all necessary data during the whole life cycle. [1]

LAN module handling the communication on the local network, which is based on managing to send and listening broadcasts. This tool is used for gathering information about swarm members. The main part is the onion domains address which is used for later important communication.

The last part mentioned in the introductory paragraph is the module arranging Tor communication. This is a very important part, as it allows a specific instance of the virus to communicate with the rest of the swarm, even in the event of disconnection from the local network, with very limited detection, because all key communication takes place via the Tor network.

When the module for communication in the secret Tor channel is needed at the first run, it's downloaded Tor project and all their necessary files for a start. The first time it needs to create the configuration before the malware starts the communication.

In configuration is needed to create Hidden Service and open (agreed) port for start server, which is responsible for receive messages from Tor network from other swarm members.

Once the Hidden service is configured, Tor automatically generates their private, public key, and hostname file containing the onion domain.

This configuration is needed only once within the first startup. Otherwise, It had to use the existing configuration to use generated onion domain.

This part of the description was only for receiving messages. For sending, it's also needed to allow parameter Socks5Listen on localhost. This listener is responsible for sending a message through the Tor network; if one of the swarm members would like to start communication, it's needed to authorize via socks5 handshake[9] and create a connection between two swarm instances. Once the sending node was connected, it's also able to send a message to another infected swarm member.

3.3 Testing and simple experiments

These experiments are focused on communication between swarm members, which is an important module for every swarm malware. This solution also offers safe communication between swarm members around the world.

[1]Both approaches are in the development and experimenting phase.

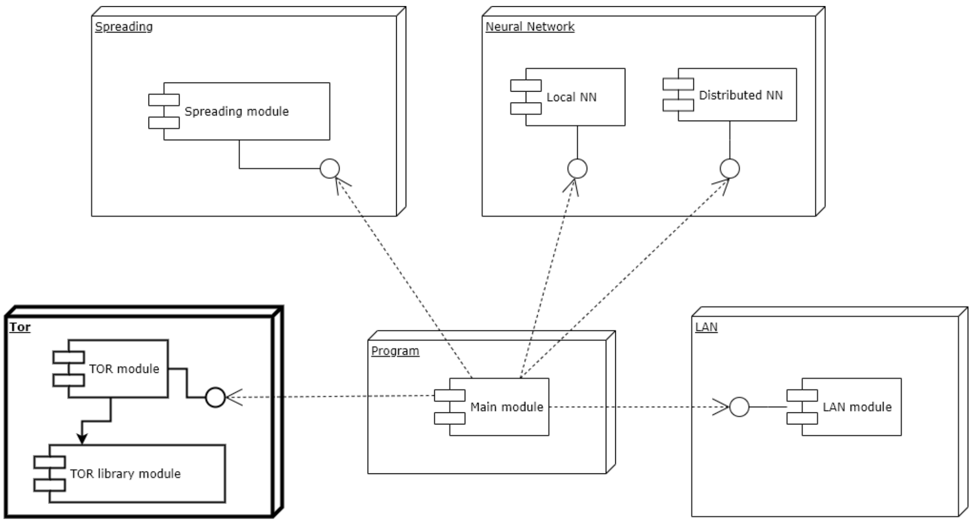

Figure 2: Malware structure

The first experiment is focused on preparing and connecting newly infected hosts into the swarm. Once the host becomes infected, it'll download TOR and create a TOR service for future communication. The broadcasts on the local network are used to let existing infected hosts in the same network know about the new member. Once the onion domain is sent as part of the broadcast message, each swarm member who receives this information will save this onion domain record.

For this scenario, 30 virtual computers were created. Each of them had fully updated Windows 10 and was connected to an internet network. Two LAN networks were used at the beginning. The experiment started with the 15 computers in the first LAN network and the other 15 computers in the second network. The computer becomes a swarm member once the malware is running. The malware was started manually, which offered us smooth network monitoring. During this testing, the DHCP server was manually manipulated, which released some member's IP addresses by force and afterwards gave them different IP addresses.

For the second experiment, two computers from the first network were moved to the second network, and different two computers from the second network were moved to the first network. After that, the four members were moved to separated, isolated, not infected networks. With this step, they were fully isolated from the original infected networks.

We hypothesize that all swarm members should be able to communicate independently.

Both of these experiments were using testing data messages with a timestamp to transfer via the tor network. Each member of the swarm virus was generating messages with timestamps and sending them to other infected nodes. The successfully received message will be used to calculate latency in the tor network.

4 Results

For the purposes of the experiment, only the installation of the Windows 10 Home and Pro operating system was chosen, in which, however, a vulnerability was manually introduced, thanks to which the malware could easily spread on the network. Malware also includes a distribution module in its architecture, which uses known system vulnerabilities, but these vulnerabilities can be fixed in future updates. So to ensure long-term repetition of the experiment, we have created our own vulnerability to be sure.

Once each swarm member downloaded the Tor and created a running service with an onion domain, the onion domains were successfully distributed across all swarm members. Each member of the swarm was able to communicate through the Tor network with the other members of the swarm independently of changing its local network.

The following table will show the latency in communication with the usage of the Tor network. Latency is calculated as the average of 100 sent messages from members in one network to members in another network or to members in the same network. According to obtained and calculated results, there is no difference in latency between local separated networks. This result was expected. Network A is our first local network with 15 computers, and network B is the second local network with another 15 computers. Networks marked as "I" are isolated networks which been created to hold one computer after the move from networks A or B.

Measured latency in communication can be slightly inaccurate but its non significant impact because of usage one local network time protocol server which was used for time synchronization across all instances.

More experiments are needed to decide which neural network approach is better and how much it is better than conventional approaches. Both approaches showed their benefits. The exact payload can be completely hidden and impossible to decode because the

Table 1: Communication between Swarm members – latency [ms]

	A	B	I1	I2	I3	I4
A	1872	1937	1995	1605	1208	1582
B	1991	1080	1378	1233	1651	1857
I1	1861	1916	1924	1344	1251	1288
I2	1683	1895	1334	1751	1811	1990
I3	1011	1472	1605	1403	1987	1147
I4	1306	1308	1554	1708	1799	1743

behaviour is hidden in a neural network. Communication is limited and not suspicious compared to synchronization of the BOT network. First experiments with a decentralized neural network across the swarm are promising from the point of resilience. To destroy the malware itself the all members must be removed with no one left because of the ability to redistribute neurons or whole parts of the neural network across the swarm.

5 Conclusion

In this article, we have introduced the use of artificial intelligence in conjunction with swarm algorithms. Artificial intelligence, in our case a neural network, was used to create a cover communication channel for the needs of swarm malware.

Both neural networks and swarm malware have been combined with stimulating the activity of malware with artificial intelligence that can move throughout cyberspace, repair its body in case of destruction and at the same time safely maintain payload and the actual communication via the Dark web. Initial experiments have shown that such hybridization of artificial intelligence and malware or antimalware of new technologies is possible and that if such a hybridization with malware is used, robust malware with very destructive power may be created. The research of the presented topic continues. In this article, we have introduced only the basic features of the communication module of the swarm virus. Currently, the network traffic of this solution is subjected to further tests.

Acknowledgement: The following grants are acknowledged for the financial support provided for this research: VŠB-TU internal grant SGS SP2021/72.

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] Anderson, H. S., Kharkar, A., Filar, B., Evans, D., and Roth, P. Learning to evade static pe machine learning malware models via reinforcement learning. *arXiv preprint arXiv:1801.08917* (2018).

[3] Bekerman, D., Shapira, B., Rokach, L., and Bar, A. Unknown malware detection using network traffic classification. In *2015 IEEE Conference on Communications and Network Security (CNS)* (2015), IEEE, pp. 134–142.

[4] Bose, S., Barao, T., and Liu, X. Explaining ai for malware detection: Analysis of mechanisms of malconv. In *2020 International Joint Conference on Neural Networks (IJCNN)* (2020), IEEE, pp. 1–8.

[5] Case, D. U. Analysis of the cyber attack on the ukrainian power grid. *Electricity Information Sharing and Analysis Center (E-ISAC) 388* (2016).

[6] Clarke, R. A., and Knake, R. K. *Cyber war.* Tantor Media, Incorporated Old Saybrook, 2014.

[7] Kolosnjaji, B., Eraisha, G., Webster, G., Zarras, A., and Eckert, C. Empowering convolutional networks for malware classification and analysis. In *2017 International Joint Conference on Neural Networks (IJCNN)* (2017), IEEE, pp. 3838–3845.

[8] Kolosnjaji, B., Zarras, A., Webster, G., and Eckert, C. Deep learning for classification of malware system call sequences. In *Australasian Joint Conference on Artificial Intelligence* (2016), Springer, pp. 137–149.

[9] Leech, M. D. SOCKS Protocol Version 5. RFC 1928, Mar. 1996.

[10] Lysenko, S. Self-adaptive method for the computer systems resilience in the presence of cyberthreads. *RADIOELECTRONIC AND COMPUTER SYSTEMS*, 4 (2019), 4–16.

[11] Mayer, M. Artificial intelligence and cyber power from a strategic perspective. *Forsvarets høgskole, IFS Insights* (2018).

[12] Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., and Nicholas, C. Malware detection by eating a whole exe. *arXiv preprint arXiv:1710.09435* (2017).

[13] Rossow, C., Dietrich, C. J., Bos, H., Cavallaro, L., Van Steen, M., Freiling, F. C., and Pohlmann, N. Sandnet: Network traffic analysis of malicious software. In *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security* (2011), pp. 78–88.

[14] Sharikov, P. Artificial intelligence, cyberattack, and nuclear weapons—a dangerous combination. *Bulletin of the Atomic Scientists 74*, 6 (2018), 368–373.

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

[16] Truong, T., Diep, Q., Zelinka, I., and Dao, T. X-swarm: The upcoming swarm worm. *MENDEL 26*, 1 (Aug. 2020), 7–14.

[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.