Swarm Inteligence in Virtual Environment

Lukáš RÉVAY and Ivan ZELINKA

Department of Computer Science, FEECS, VŠB - Technical University of Ostrava, 17. listopadu 15, 708 33, Ostrava - Poruba

Corresponding Author: Lukáš RÉVAY(email: lukas.revay.st@vsb.cz)

(Received: 3-May-2019; accepted: 28-June-2019; published: 30-June-2019)
DOI: http://dx.doi.org/10.25073/jaec.201932.242

Abstract. *To simulate some behavior of swarms, malware was selected as carrier of intelligence. This article describes current solution which is fully virtual. This gives us possibility to interfere environment and see how the improved malware will react. This common intention provides improvements related to docker images and also architectural that is related to code changes. Communication over network together with cooperation on particle level is key part of this solution. Malware movements are same as movements of swarm particles, which fully fit this requirement. Significance is also put on swarm part, where the decision which swarm algorithm to utilize is crucial. Outcome from this work should be partly practical and theoretical related to environment setup, particles communication, movements and coordination which finally finishes in distributed denial of service (DDoS) coordinated attack via hypertext transfer protocol (HTTP) to some server. After this theoretical work the practical simulation will be done to see if the swarm attack brings expected results.*

Keywords

Docker, PSO, Swarm, S-bot, Linux

1. Introduction

This work seamlessly follows and extends the malware based environment which was sit up in previous work [1]. Additional implementations like sensors and network communication are practical. Still there are ongoing implementations and because of that the theoretical concepts are ahead of implementations. Theoretical idea of the swarm based network malware is the added value of this paper. This is the main reason why this work is done and all related stuff which has been already implemented fully supports this theoretical concepts. To make this work understandable and also ready for later testing the main scenario for this work is prepared. The primary aim is the DDoS attack to the HTTP server. It is a coordinated brute forced request attack. To do so a couple of prerequisites related to swarm (movements, collision detection, etc.) and communication was necessary to solve. But let's return to beginning before it all starts. The aim was to choose and integrate swarm algorithm and do movements over the network with all necessary communication. Communication is described in chapter 3, especially part for connection-less setup which is implemented. Because network is not friendly environment stuff related to collision detection (particle distance to another particle or particle collision detection) is introduced in chapter 6.1. It can also solve problems related to obstacles like firewalls or computer stop that can make particle unresponsive. As was mentioned previously this is still on a theoretical level. To fully prove the working swarm malware the practical implementation needs to be finished.

At the end it is also mentioned how the future work will continue. It is important to follow such plan. This paper is part of a bigger solution which will be compound from separate papers in the future.

2. Current situation in swarm malware algorithms

The term “virus” means a self-replicating structure that uses host objects (as file or PC in the case of computer worm) to make its copies and attack new hosts. It often refers to any infiltration, regardless of whether it is a virus. The malware (for this paper purposes we understand it as a computer virus if not stated differently) was described theoretically as a study based on Turing machines, Von Neumann cellular automata and self replicating program, including Langton’s automata. The pure existence of such program is fully dependent on recursive function [2], halting problem [3] which generally describes self-reproducing automata [4].

Word virus is not related only to computer science. To be precise, the virus notion had been used in biology before computer science. But the similarities with respect and understanding of viruses is also part of the IT world. It cannot survive without any host. In the world of computers connected with networks, it has a lot of possibilities for execution which finally gives the base for self-replication. Host objects like executable files, system area on disk or files run by specific applications (Libre office, Microsoft Word, shell scripts, etc.) provide environment for virus. After execution of such file virus becomes active and starts to do self-replication to another attachable host object.

Virus simply "infects" other programs by including it’s copy to some part of the target(program, executable etc.) [5]. After successful infiltration into PC the infected program in some particular time executes also the virus. The aim of the virus is to spread over whole PC, network and after that do some harm or just spy. There are various types of malware. It depends what is their aim and how they attack objects of interests. For worms the typical environment for spreading is the network and the host objects are PCs in this infiltrated network. Botnets as a remotely controlled malware from one server and trojan horses among the others [6].

Centralised control of viruses makes them an easy target for elimination. Modern anti-malware systems can adapt to the newest threads. Utilizing artificial neural-networks, artificial immune system algorithms [7–9] makes a good defense against such threads. Let’s call this kind of malware as a classical one. Such type is single program creating self copies. The weakest part of such malware like botnet is the centralised control from one point. Centralisation makes the virus vulnerable when the anti-malware systems detect it. It can then block open ports or do the forcible stop of the program which runs the malware inside. Because anti-malware systems are highly adaptive with AI supportive algorithms, as was stated earlier, it is not striking that also malware evolution and development is still ongoing.

The increasing complexity of malware algorithms moved the solutions to evolutionary techniques, artificial intelligence as well as swarm intelligence [4]. This kind of decentralisation increases the robustness of such solution. Future will bring threats related to unknown malware, which shall be the reason why the environments for testing and analysis of such viruses should begin. Many optimisation problems have been solved by swarm algorithms that designate those algorithms also for malware purposes. Without any leader or control point, the whole swarm together can act and solve problems according to its intelligence [10]. To get rid of any harm on real computers and still have the possibility of controlled simulations provided by virtualized environment [1].

2.1. Inside virtual world

Current work continues with virtual environment which is backbone infrastructure for swarm malware development and testing. It allows multiple users to share improvements via docker images without influencing each other on architectural or even code level [1]. Such kind of separation of a virtual environment and real connected computers allows making such attacks without any harm or even impact in real computers.

Alpine based image was sit up as an environment which fully supports program execution. To prepare such image it was defined which tools are needed to fit the needs. These tools have been added into image as illustrated here. Also libraries which are required as prerequisites for program run are parts of this Fig. 1.

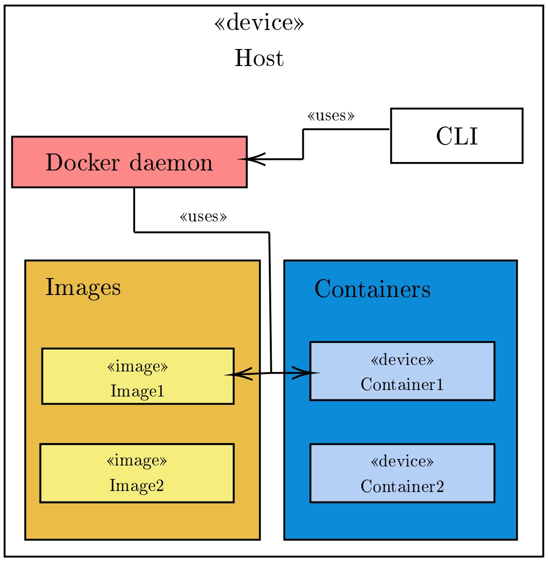

Fig. 1: An illustration representing hosting of docker artifacts together with command line interface (CLI) used for control

2.2. Virtual networking

Virtualization itself brings benefits mentioned above. But there are a couple of problems like the network topology. In real network there are active network devices like routers, switches, computers and others. In this solution from routing point of view there are no any devices which can split the network on segments or do routings. That makes each node one hop far from another one. This is not fully suitable for DDoS attack where the movements are necessary to bring the particles as near as possible to the target node.

To mimic the network like a real one the solution is to set up firewalls. It means that the particle will not be allowed to do the copy of itself to all nodes but only on some of them. To close these ports of some of these computers make some subset of non-usable nodes for particles. To accomplish also the need of DDoS attack some nodes cannot make HTTP requests to a target node. This creates another subset of non-usable particles for attack.

3. Network cooperation

Crucial part of this work is network communication. To see how the swarm is behaving we decided to put it on fully connected virtual network. This solution allows us to see how whole swarm will react on network changes. Let's imagine that there is swarm malware. Each particle occupies some virtual node in network. The aim of the swarm will be to do the DDoS attack on HTTP server. That means that in one particular moment they will do HTTP requests to this server. Before the attack starts each particle should be on its place where the HTTP server is reachable. To make the successful attack as in Fig. 2 the number of requests must be enormous. So when these requests are done in parallel the likelihood that the attack will succeed is much bigger.

The swarm is doing the attack which is the primary aim. All communication is done via network. There are a couple possibilities of how the particles can communicate. Next chapters will describe possible communications setups.

3.1. Connectionless setup

This setup is currently utilized in practical part. To introduce this solution it is necessary to mention that user datagram protocol (UDP) packets are important for this. How to inform other particles over the whole network? The right answer is, to broadcast packets with important data. These data are then red by all particles. Each particle can send and also receive such packets.

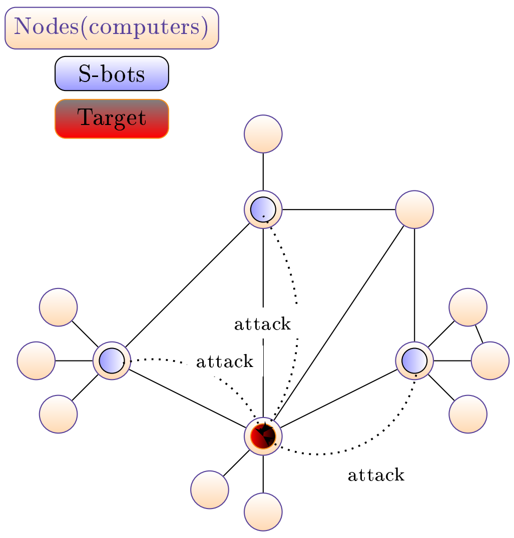

Fig. 2: Particles coordinated attack on http server

Connectionless communication between particles does not make direct connections to each other. It means no established connection, and what is most important, no initial setup of connection between particles. According to previous sentence also the load on network can be minimized because communication happens periodically. The drawback of this can come in not guaranted delivery, where the messages can be lost.

Loose coupling between particles makes the particles more independent. But the drawback of this architecture is how to synchronize them and share the information correctly in time. That means there must be no collision during message broadcasting.

3.2. Connection-oriented setup

This kind of setup is still theoretical. It hasn't been implemented yet. But still it is necessary to count with this solution which can bring another insight. In this case this solution will need swarms which particles are tightly coupled and coordinate in groups or even one big group.

3.3. Combined setup

Previous two solutions give another possibility where some improvements can be done. Imagine that group of three articles is tightly coupled. There will be a couple of such groups which will communicate without any direct connection, only via UDP broadcasting where each group will have a master. Masters from each group will coordinate the global aim swarm has.

The solution can be also inverted compared to first paragraph. That means that tight coupling will be between masters and loose coupling between particles in the subgroup.

4. Communication messages

To make particles communicate and cooperate together, it was necessary to setup messages which are understood by particles and can carry all necessary information. Each message contains information mostly connected with a swarm. From swarm perspective point of view the message must contain:

- G_{best} (best position in swarm)
- cycle number (number of iteration)
- particle id (particle which sends current message)
- next particle id (particle which is going to send in next iteration)
- not usable addresses

Each particle is able to read and also send this kind of message. Because particles are not connected directly it is very important to synchronize message sending with the background running algorithm providing the backbone of this solution.

5. Swarm utilization

Based on previous chapters about particle network communication (chapter 3) it was necessary to decide which swarm algorithm will be used for this setup. The prime intention was pointed to the loosely coupled swarm particles. That means each particle will solve the moving problem via network until it is close enough the HTTP server. Each particle should than know that is on a place and also that other particles are on place (zero hops). This restriction setups algorithm end condition.

There are many swarm algorithms or those modifications which are hot candidates for this work. More specifically it was thought about grey wolf optimisation (GWO) [11], particle swarm optimisation (PSO) [12], ant colony optimisation (ACO) [13]. Each of previously mentioned algorithms brings another specific solutions to optimisations. Because this work shall be a proof of concept for incoming solutions it was decided that for simplicity the PSO will be enough. Of course in future these solution can be improved and another algorithms can be utilized.

PSO is appropriate algorithm because of mostly caused by the easiest way of its implementation. Implementation itself shall obey practices which can later provide easier changes of algorithms. The inspiration comes from optimisation framework which this implementation is inspired. Changes related to abstraction needs and object oriented programming (OOP) patterns are changed a bit to fulfill precisely needs of this work.

It is clearly stated in that OOP is a great way to create swarm intelligence [10] based systems easily because the particle philosophy often directs researchers to design operation elements with features and functions as similar to the mechanisms within OOP. This approach was applied while developing this solution [14].

5.1. Particle swarm optimisation framework

Particle swarm optimisation (PSO) is a population based stochastic optimisation technique developed by Dr. Eberhart and Dr. Kennedy in 1995 inspired by social behavior of bird flocking or fish schooling. The particle swarm concept originated as a simulation of simplified social system. Its original intention was to graphically simulate the choreography of birds or fish school. However, it was found that particle swarm model can be used as an optimizer. PSO is often used to find the maximum or minimum of a function. Although it is generally used in static problems, PSO has also proved useful in dynamic problems, where the environment changes [12].

This algorithm adopts animals technique when finding food in groups. Every individual in PSO is assumed as a particle. Each individual is trying to optimize its fitness function by exploring the search space using its own information as well as information from other individuals. Each individual must keep track of its best position which is the position with the highest fitness function value perceived. This position is called the personal best or local best value.

PSO algorithm is convergent where all particles at a particular iteration will eventually wander around global best position. If this happens, then the movement of those particles would not be too significant hence make PSO computation becomes ineffective. At this point decision will be made whether search will continue or not. This determines whether the convergence properties found a solution that can already be considered successful or not. Whole solution is more robust. This is caused by communication over network where is necessary to synchronize and inform particles about G_{best} [12] position in whole swarm bot (more about swarm bot 6.). This information will be used in next iteration.

In this simulation attractiveness is than the precise HTTP server port which has the higher priority than another open ports. Objective function is not so complex even multimodal. PSO can be utilized because of particles and sensors data from s-bots (more about s-bot 6.1.), where each particle has its velocity according to the equation

$$\begin{aligned} v_i(t+1) = v_i(t) + c_1.r_1(P_i - x_i(t)) \\ +c_2.r_2(G_{best} - x_i(t)) \end{aligned} \tag{1}$$

and position

$$x_i(t+1) = v_i(t+1) + x_i(t) \tag{2}$$

Each particle computes it's own fitness. Environment where particles move happens is discrete and finite [15]. This space is separated into groups related to sub-networks 6.3. based on IP addresses. Whole swarm is able to move via such space and share important information. Discrete character of this environment changes the equations a bit. It is not surprise that we can get rid of $x_i(t)$ (in equation 1) part because it makes no sense to count the fitness based on IP address. This also gives us answer that whole position is totally secret and cause by that it is not necessary to use this equation 2. What is still important for the solution is inside objective function.

5.2. Objective function

The task int this work was to find the HTTP server in network. Bring the particles as near as possible to this node. What does it mean near in network domain? In this case it means one hop far from this target is the nearest position. Each particle shall be on this position. If this is not possible, caused by higher number of particles than zero hop distance nodes, than the remaining particles will stay on one or n-th level of hop counts. The best positions of all particles are when all are a the sub-network where a target exists. It can be easily proven that all particles zero hop far are those which are on the same sub-network.

To make it general let say that the objective function will take into account number of open ports or specific port and also hops to target. Let say particle which finds a node with ten open ports on target node one hop far, will have better position than particle with five founded open ports and one hop far from target node. It is clear that the objective function 3 counts only with discrete values related to ports and hops, but for this demonstration ,academic purposes and further investigation this is enough.

$$P = open\_ports\_count - hops\_count \tag{3}$$

Objective function takes care only about hops count and open ports. Of course it can happen that prematurely searching ends in starting sub-network(network where swarm is initialised). This can be easily avoided when the algorithm will be changed a bit in way that will not stop with DDoS attack. There can be also another improvement that during infiltration on system the virus can scan for specific files where it can gather information(hosts file, routing table, known hosts in ssh, etc.) for movement into another network.

6. Swarm bot

Thinking about movements of the particle in any space brings the same problems which needs to be solved in space of network devices. Similar solutions are already done in swarm bots [16] solutions. Because the difference is just in domain where this is used it was decided to apply some parts of this solutions also in this work.

Each s-bot is a fully autonomous mobile robot capable of performing basic tasks such as autonomous navigation, perception of the environment and grasping of objects. In addition to these features, an s-bot is able to communicate with other s-bots and physically connect to them in flexible ways, thus forming a swarm bot [16].

As was stated in previous paragraph s-bots make physical connection. This is possible in real world environment but not in the network virtual one. It can be mimicked for example by connection oriented setup as was mentioned in the chapters 3.2 and 3.3.

6.1. S-bot

S-bot is a fully autonomous robot and is equipped with all the sensors necessary for navigation and movements [16]. Environment the bots are operating in this work is not so so general for movements. This causes a different usage of sensors.

In this domain sensors like cameras or infrared sensor are not necessary. Tools like ping, telnet, traceroute, finger and etc. are sensors which can be utilized in this domain related to IP addresses space 6.3. .

There are used words like particle and s-bot in this paper. To make it clear, let's assume that s-bot and particle are synonyms when it comes to swarm intelligence particle is more precise. In case of swarm bot and self intelligence the s-bot will be used.

6.2. Particle move

Particles movements 3 are crucial prerequisites for swarm bot. To keep the basic principles of movements was the important part. Movement of each particle obeys rules which are necessary for swarm bot as mentioned in chapter 6. The movement itself consists of few steps until it happens.

States of particles like copy and then delete do the needed job. But before delete the particle have had to start another one. Move state does the movement to destination. This kind of setup provides all unnecessary stuff which fulfills requirements of swarm bot.

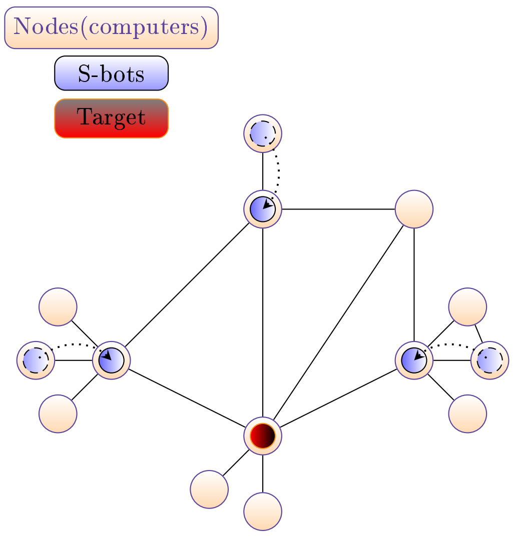

Fig. 3: Particles move in virtual network

6.3. Space of addresses

How to easily move the particles over network? Well there are couples of methods how to do so. In network the only heading is the IP address which fully suites swarms. To find the shortest path the standard tools like trace-route or ping can be utilized. Each particle chooses address according to basic rule it obeys as a swarm bot as is mentioned in chapter 6. Cognitive part and also swarm part are counted into decision which node will be selected next.

The computation of new address will change according to inputs, like potential node exists and listens on particular port. Checks if the address is suitable can be part of swarm algorithm or as a post check after each iteration. This decision directly impacts the results of performance of the swarm.

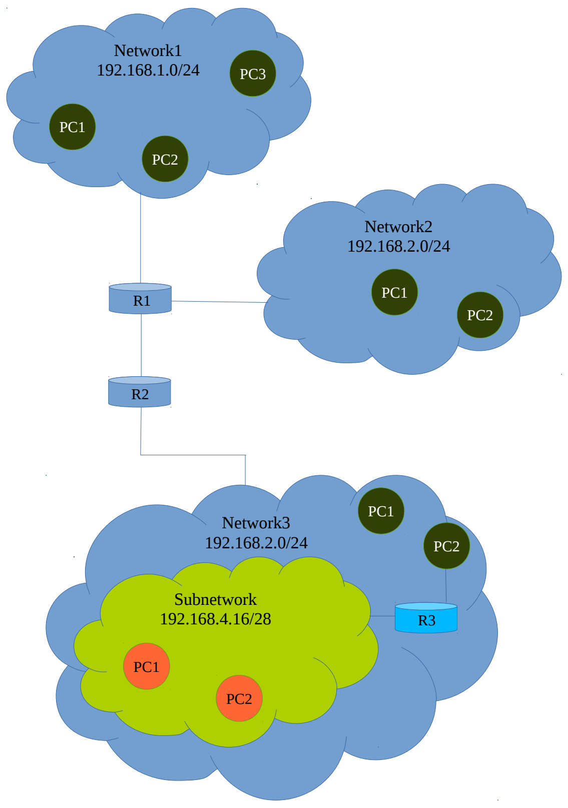

Fig. 4: Network and subnetwork example with nodes

The movement of whole swarm is fully dependent on unique identification of each node in network. This is provided by IP address assigned to each node in network. Whole address space defined by four dot separated numbers provides also subnetworks. Each subnetwork is defined by its network address and a netmask. This separates whole universe of one network to multiple networks and subnetworks. This solution is counting with this and provides s-bot(particle) movements via this multinetwork environment.

Most important on this is how to count the distances, which is crucial part of swarm algorithm for this solution as described here (chapter 5.2) is how to measure the distance between each node which is based on a network distances as described in the table (1) below for networks (see Fig. 4).

	Network1	Network2	Network3
Network1	0	1	2
Network2	1	0	2
Network3	2	2	0

Tab. 1: Distance table for networks related to Fig. 4

This observation can be applied also on each computer(node) which belongs to particular network and wants to communicate with another computer. Such observation is a main setup when it comes to hops count during particle movement.

6.4. States of particles

Most important state the particle goes through is the movement state. It is because of the utilization of PSO as swarm algorithm. This state is necessary for PSO algorithm in case the particle moves through network. The movement is computed from IP addresses set related to the network. Movement itself is done by two steps. In first step the malware must copy to selected node. After the copy is on place it must be activated. After success activation the old malware is stopped and deleted so that there will be no evidence of it on node anymore. On the way they could be addresses not assign to any node or nodes which are not allowed to accept ssh session. In that case the malware will stay on the current node.

When does it all end? For this particular solution it ends when the particles are on place. It means that all particles are few hops from the target or all are in same sub-network. At some point the particles synchronize over whole swarm that there are no more better places to do the attack. And this is the time when it all starts. Particles start requesting some HTTP resource from server concurrently from all positions and that is a attack state as displayed on the Fig. 5.

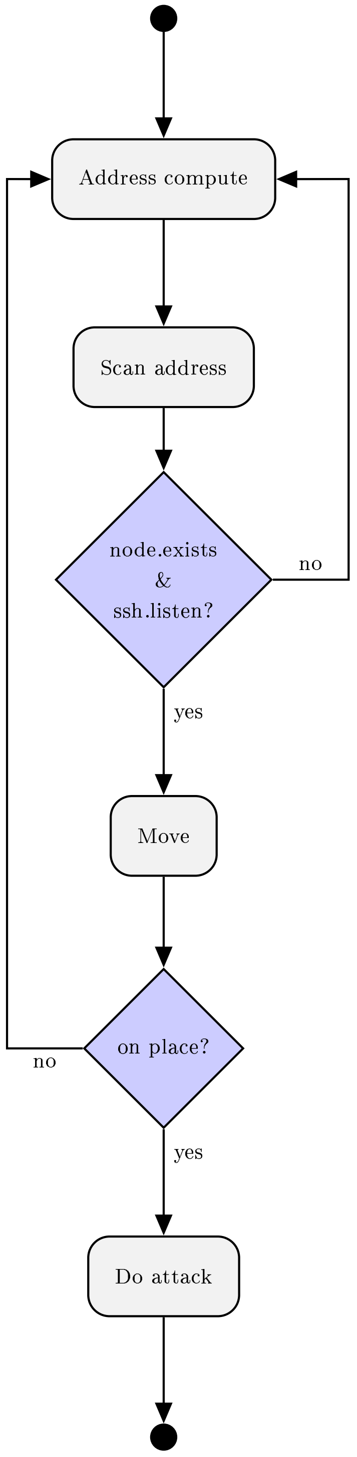

Fig. 5: Describing the state of particle

7. Future improvements

Future improvements should add possibilities to spy particles communication on real time, to see which particle has for example the best solution, how the swarm reacts when topology of network changes, when some particle from swarm is not reachable and other changes related to firewalls not running ssh's blocked ports and so on.

Crucial will be also the improvement of swarm algorithm. This improvement shall bring possibility to swarm dynamically reacting on changes as was mentioned in previous sentences. There are couple of improvements for future work like target selection for DDoS attack, encryption of sent messages during communication and coordination, random sending of messages in swarm and also connection(less) implementation mentioned in chapter related to this topic. These all improvements can make the platform more robust and can bring different approaches visible from testing.

Let's come back to architecture. It will also be helpful to create API for swarms where it will be easy to change the algorithm which is responsible for this intelligence part as mentioned in chapter 5. The API shall allow to test the algorithm part via unit tests which can spare a time in future development. The parallelization of currently sequence running tasks will be also on place to make the malware spreading faster and more performance proven.

This solution must not be used only for DDoS attacks, but also for detection of viruses, fragile part of infrastructure searches and stress testing of environment. All these possibilities can be achieved with swarm algorithm changes or objective function changes that can make bigger insight to this problematic.

8. Summary

Aim of this work was to find possible solutions for further implementations and partly implement swarm intelligence and make the separation of components for better testing, demarcation of responsibility and future improvements. Crucial thing to solve was also the network setup in virtual environment described in chapter 3.1. Most of these problems are solved theoretically. In particular time these theoretical solutions will become part of the platform implementation.

Acknowledgment

The following grants are acknowledged for the financial support provided for this research: Grant of SGS No. SP2019/137, VSB Technical University of Ostrava.

References

[1] Révay, L. (2019). From Malware Testing to Virtualization. *Procedia Computer Science*, *150*, 751 – 756, proceedings of the 13th International Symposium "Intelligent Systems 2018" (INTELS'18), 22-24 October, 2018, St. Petersburg, Russia.

[2] Shepherdson, J.C. & Sturgis, H.E. (1963). Computability of Recursive Functions. *J ACM*, *10*(2), 217–255.

[3] J.E. Hopcroft, J.U., R. Motwani. *Automata theory, languages and computation.* International Edition 24.

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

[5] Cohen, F. (1987). Computer viruses: Theory and experiments. *Computers Security*, *6*(1), 22 – 35.

[6] Szor, P. (2005). *The Art of Computer Virus Research and Defense.* Addison-Wesley Professional.

[7] Forrest, S., Hofmeyr, S.A., & Somayaji, A. (1997). Computer Immunology. *Commun ACM*, *40*(10), 88–96.

[8] Kim, J. & Bentley, P.J. (2002). Towards an artificial immune system for network intrusion detection: an investigation of dynamic clonal selection. In *Proceedings of*

the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600), volume 2, 1015–1020 vol.2.

[9] Zhao, G., Zhang, C., & Zheng, L. (2017). Intrusion Detection Using Deep Belief Network and Probabilistic Neural Network. In *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, volume 1, 639–642.

[10] Eberhart, R.C., Shi, Y., & Kennedy, J. (2001). *Swarm intelligence*. Elsevier.

[11] Kohli, M. & Arora, S. (2018). Chaotic grey wolf optimization algorithm for constrained optimization problems. *Journal of Computational Design and Engineering*, *5*(4), 458 – 472.

[12] Hu, X. & Eberhart, R.C. (2002). Adaptive particle swarm optimization: detection and response to dynamic systems. *2*, 1666–1670.

[13] Blum, C. (2005). Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, *2*(4), 353 – 373.

[14] Mani, N., Helfert, M., Pahl, C., Nimmagadda, S.L., & Vasant, P. (2018). *Domain Model Definition for Domain-Specific Rule Generation Using Variability Model*. Cham: Springer International Publishing.

[15] Sabir, M.R., Mian, M.S., Sattar, K., & Fahiem, M.A. (2007). IP Address Space Management using Aggregated Fixed Length Subnet Masking. In *2007 International Conference on Electrical Engineering*, 1–4.

[16] Nolfi, S., Denebourg, J.L., Floreano, D., Gambardella, L., Mondada, F., & Dorigo, M. (2003). Swarm-Bots: Swarm of Mobile Robots able to Self-Assemble and Self-Organize. *Ercim News*, *53*, 25–26.

About Authors

Lukáš RÉVAY is a doctoral student of VŠB-TU Ostrava University. His main area of research is swarm inteligence utilised in cyber security. His recent activities include virtual environment setup for testing purposes, development of swarm malware, and malware monitoring.Lukáš is primary focused on swarm intelligence. He is also keen on data analysis which he wants to combine later with swarm malware and use it for data collecting and statistical purposes related to malware swarm behaviors. He works as an software engineer which brings him a lot of insight to problematic related to cyber security when he is working on commercial projects for a various of clients. Marked by these experiences he started to think about another threats that started his interest on swarm malware and malware detection. Lukáš likes to learn new things from different domains and utilise those knowledge in other parts of life when possible. He lives in Ostrava, Czech Rep., is interested in sports and likes motorbikes.

Ivan ZELINKA is currently working at the Technical University of Ostrava (VŠB-TU), Faculty of Electrical Engineering and Computer Science and national supercomputing centre IT4Innovations.He also has been invited for lectures at universities in various EU countries as well as for keynote and tutorial presentations at various conferences and symposia. He is responsible for supervising the research grant from the Czech grant agency GAČR named Soft computing methods in control and was co-supervisor of the grant FRVŠ - Laboratory of parallel computing. Dr. Zelinka has also participated in numerous grants and two EU projects as a member of the team (FP5 - RESTORM) and as supervisor of (FP7 - PROMOEVO) of the Czech team. He was awarded the Siemens Award for his Ph.D. thesis and received an award from the journal Software news for his book about artificial intelligence. Ivan Zelinka is a member of the British Computer Society, IEEE (a committee of Czech section of Computational Intelligence), and serves on international program committees of various conferences and three international journals (Soft Computing, SWEVO, Editorial Council of Security Revue). He is the author of numerous journal articles as well as books in Czech and the English language.