Securing computers networks from malicious worms spreading using an Influence Maximization based solution

Tom Landman, Tamar Levy

Introduction

In reality, nowadays there is an enormous number of computer networks and cyber threats respectively. One of the major cyber threats for a network is computer worms that can spread within a network and infect a large number of hosts and servers and harm end-users and organizations in multiple ways. In this research, we aim at creating an Influence Maximization based solution to secure computers network from malicious worms spreading, as well as getting an understanding regarding the network and nodes features that might affect the worms spreading.

In the *Background* section, we present information regarding computer networks, cybersecurity, computer worms, and servers. The problem which we cope with is defined at the *Problem Definition* section. The *Methods* section is divided into multiple subsections. In this research, we simulate a large computer network that contains two sub-networks, hosts, servers, security components, etc. all the information regarding this issue is elaborated in the *Data Generation* subsection. the output of this phase in our research is a data frame with information regarding the network created, and the network's graph. After the data was generated, we used an Influence Maximization - Independent Cascades Model (ICM). The ICM receives the origins of the infected nodes and outputs the influence the worm's spreading within the network. The whole process is detailed in the *Influence Maximization – Independent Cascades* subsection. The ICM is problematic since for large networks there can be many combinations of origin nodes to be tested in order to cover the whole solution space in a reasonable amount of time. Hence, this is a classic Non-Polynomial (NP) Hard problem. In order to gain a sufficient approximate to the optimal solution in a reasonable amount of time, we used the Memetic Algorithm (MA) combined with the ICM, as elaborated more in the *Memetic Algorithm* subsection. The MA combines both global and local search methods that ensure a better convergence to the optimal solution space. In the *Random Forest Regressor* subsection, one can find information regarding the final phase of our research. We created a Random Forest Regressor to predict the probability of nodes to be infected by the computer worm. We also present a feature importance plot of the regressor model to assist security experts to get an understandable picture of the nodes and networks' characteristics that influence their protection from the worm infection. The *Results* section contains our research results.

Background

In this section, we discuss both theoretical and practical concepts that are required for a better understanding of why and how we implemented our proposed system. First, we discuss Computer Networks to emphasize how computers communicate with other computers within a network. Then, since these computer networks became an attractive target for hackers, we explain some key concepts regarding Cyber threats and Cybersecurity.

i. Computer Networks

The merging of computers and communications has had a profound influence on the way computer networks are organized. Computer Network is a collection of autonomous computers interconnected by a single technology, whereas two computers are said to be interconnected if they can exchange information [1]. The connection does not need to be via a copper wire since fiber optics, microwaves, infrared, and communication satellites can also be used. Networks come in many sizes, shapes, and forms, and they are usually connected to make larger networks, such as the Internet which is the most well-known example of a network of networks.

From a business perspective, most companies have a substantial number of computers. For instance, a company may have a computer for each worker to provide a suitable environment, services, and software needed for its tasks based on different types of roles. Initially, some of these computers may have worked in isolation from the others, but at some point, management may have decided to connect them to be able to distribute information throughout the company and to support physical resource sharing such as Printers. Both small and medium businesses (SMBs) and large corporates are vitally dependent on computerized information. For instance, most companies have customer records, product information, inventories, financial statements, tax information, and much more online. Hence, a scenario in which all computers suddenly went down can majorly harm and effect such companies from several aspects such as financial, reputational, etc. As a result, companies are exposed to threats, cyber-attacks, and become a coveted target for hackers and therefore, should invest time and capital to mitigate these cyber threats which will be elaborated on in the next section.

In the next subsection, we discuss and define important key concepts regarding computer networks that are crucial for understanding our proposed solution discusses in the methods section.

1. Open Systems Interconnection model (OSI model)

The OSI model is a conceptual model that characterizes and standardizes the communication functions of a telecommunication or computing system without regard to its underlying internal structure and technology [1]. As can be seen in Figure 1, the communications between a computing system are divided into seven abstraction layer architecture, each of which is responsible for performing specific tasks concerning sending and receiving data. Also, note that we will focus on the Network Layer and Transport Layer since they are much more relevant for our proposed method.

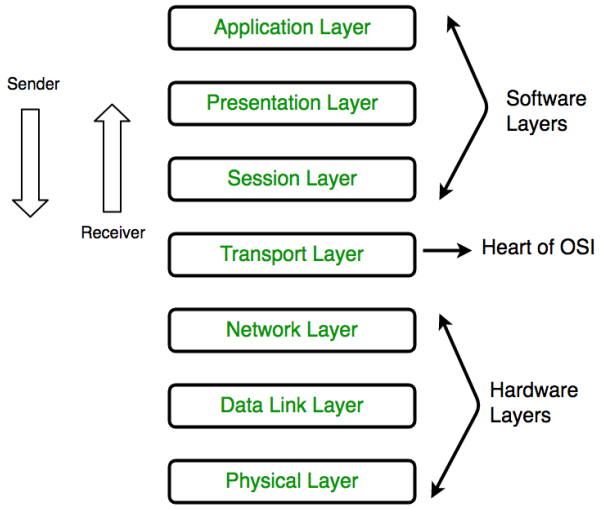

Figure 1. OSI Model's layers.

This 1st layer, denoted as *Physical Layer*, consists of the physical equipment involved in the data transfer, such as multiplexer boxes, network hubs, cabling, repeaters, network adapters, modems, etc. The physical layer is responsible for the actual physical connection between the devices. This is also the layer where the data gets converted into a bitstream, which is a string of 1s and 0s. The 2nd layer denoted as *Data Link Layer* provides reliable and efficient communication between two or more devices. The main function of this layer is responsible for the unique identification of each device that resides on a local network. The 3rd layer, known as *Network Layer*, mainly responsible for facilitating data transfer from one host to the other located in different networks. This layer finds the destination by using logical addresses denoted as Internet Protocol (IP) addresses. IP address distinguishes each device uniquely and universally. Further, routers are a crucial component used to quite literally route information where it needs to go between networks. The 4th layer denoted as *Transport Layer* can be called as an end-to-end layer as it enables a point-to-point connection between source and destination to deliver the data reliably. The data in the transport layer is referred to as Segments. This layer is also liable for flow control and error control. The 5th layer e.g. *Session Layer* used to establish, maintain, and synchronize the interaction between communicating devices. This layer assures that the session stays open long enough to transfer all the data being exchanged, and then promptly closes the session to avoid wasting resources. The 6th layer denoted as *Presentation Layer* is also called the translation layer formats or translates data for the highest layer based on the syntax or semantics that the application accepts. This layer is in charge of translation, encryption, and compression of data. Last, the 7th layer known as the *Application Layer* allows the user to interact with the application or network whenever the user elects to read messages, transfer files, or perform other network-related activities. Application layer protocols include Hypertext Transfer Protocol (HTTP), as well as the Simple Mail Transfer Protocol (SMTP), which is one of the protocols that enable email communications.

2. TCP/IP Model

The Transmission Control Protocol/Internet Protocol denoted as the TCP/IP model [1] was designed and developed by the Department of Defense (DoD) during the 1960s

and is based on standard protocols. This model is a concise version of the OSI model explained above, however, as can be seen in Figure 2 below, contains only four layers, unlike seven layers in the OSI model.

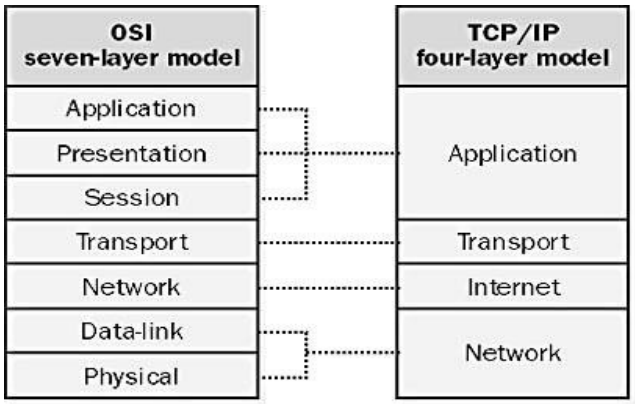

Figure 2. Comparison between OSI & TCP/IP Models.

Note that from the receiver perspective the 1st layer is the Network Layer which corresponds to the combination of Data Link Layer and Physical Layer of the OSI model. It looks out for hardware addressing and the protocols present in this layer supports the physical transmission of data. The 2nd layer denoted as the Internet Layer parallels the functions of OSI's Network layer. It defines the protocols which are responsible for the logical transmission of data over the entire network. **The main protocols residing at this layer are:**

- **Internet Protocol (IP)** is responsible for delivering packets from the source host to the destination host by looking at the IP addresses in the packet headers. Moreover, as can be seen in Figure 3 .below, IP has 2 versions: IPv4 and IPv6. IPv4 is the one that most of the websites are using currently. However, IPv6 is growing as the number of IPv4 addresses is limited in number when compared to the number of users. Note the difference between IPv4 header structure and its length of 32 bits that is different from the IPv6 header, which is composed of 64 bits, hence, supports a wider range of addresses.

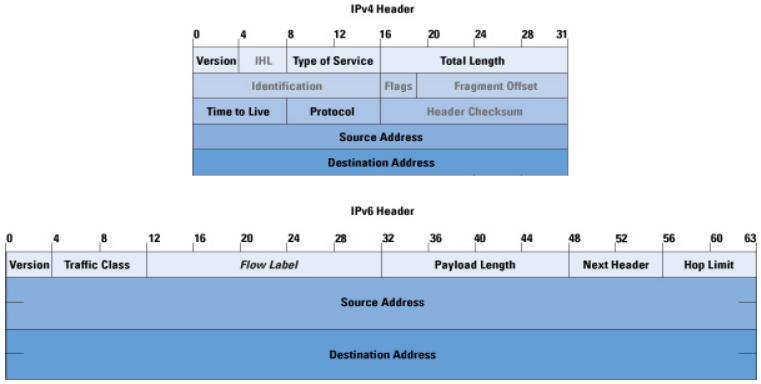

Figure 3 . Comparison between IPv4 and IPv6.

- **Internet Control Message Protocol (ICMP)** is encapsulated within IP datagrams and is responsible for providing hosts with information about network problems.
- **Address Resolution Protocol (ARP)** is used to find the hardware address of a host from a known IP address. ARP has several types: Reverse ARP, Proxy ARP, Gratuitous ARP, and Inverse ARP.

The 3rd layer, known as the Transport Layer, is analogous to the transport layer of the OSI model. It is responsible for end-to-end communication and error-free delivery of data. It shields the upper-layer applications from the complexities of data. **The two main protocols present in this layer are:**

- **Transmission Control Protocol (TCP)** provides reliable and error-free communication between end systems. It performs sequencing and segmentation of data. It also has an acknowledgment feature and controls the flow of the data through its flow control mechanism. It is a very effective protocol but has a lot of overhead due to such features. Increased overhead leads to increased costs.
- **User Datagram Protocol (UDP)** is the go-to protocol if your application does not require reliable transport as it is very cost-effective. Unlike TCP, which is a connection-oriented protocol, UDP is connectionless.

The last 4th layer denoted as Application Layer performs the functions of the top three layers of the OSI model: Application, Presentation, and Session Layer. It is responsible for node-to-node communication and controls user-interface specifications. Some of the protocols present in this layer are HTTP, HTTPS, File Transfer Protocol (FTP), Secure Shell (SSH), etc.

3. Subnet Mask

As mentioned above, an IP address is similar to a postal address. Whether it is a postal address or an IP address, it contains two addresses, group address, and individual address. In a particular group, the group address is common for all hosts and the individual address is unique for each host. Similar to the postal system where group address is known as area address and house address represents the individual address, in the IP network, these addresses are known as network address and host addresses respectively. In an IP address, the number of bits that are used in a network address and the number of bits that are left for the host address are determined by a subnet mask. As can be seen in Figure 4 below, for an IP address of format IPv4\IPv6, a subnet mask is also 32\64 bits long address respectively and can be written in both binary and decimal notations. The subnet mask is used to distinguish the network portion from the host portion in an IP address. Therefore, when software needs to determine the network ID portion of a given IP address, it performs a logical AND operation with the given IP address bits and the subnet mask bits. There are 4,294,967,296 IP addresses in total which are categorized into five classes: A, B, C, D, and E. As can be seen in Figure 5 below, these classes mainly differ from each other by the number of networks and the number of host addresses per network.

Subnet Mask 255.255.255.0				
	24 bits for Network ID			8 bits for Host ID
Decimal	255	255	255	0
Binary	11111111	11111111	11111111	00000000

Figure 4 . Subnet Mask in IPv4 format.

For instance, in the case of multiplying a given IP address that belongs to class C, e.g. 192.168.10.11 by its suitable subnet mask of 255.255.255.0 (/24 means that 24 bits are assigned for the number of blocks), the resulted address would be 192.168.10.0. Hence, the 24 bits prefix is equal, but the last 8 bits can be modified between (0, 255). Also note that for class C, any IP address in the given range has a subnet mask that can be

used to create a network portion of approximately 2.1 million addresses and a host portion of 256 addresses.

The given range has 256 different combinations that a host's IP address can potentially be assigned to. However, the first and last numbers, e.g. 0 and 255, are used in all networks for the network address and broadcast address and therefore are not available for node addressing. So, since there are 256 addresses in the network, there can only be 254 nodes defined.

Every IP Addresses in the Internet		Class	Classful IP Ranges	Subnet Mask for each Block	Number of Blocks	IP addresses per Block
0.0.0.0 /0	Unicast	A	**0**.0.0.0 - **127**.255.255.255 0.0.0.0 /1	**255.0.0.0** **/8**	128	16,777,216
		B	**128**.0.0.0 - **191**.255.255.255 128.0.0.0 /2	**255.255.0.0** **/16**	16,384	65,536
		C	**192**.0.0.0 - **223**.255.255.255 192.0.0.0 /3	**255.255.255.0** **/24**	2,097,152	256
	Multicast	D	**224**.0.0.0 - **239**.255.255.255	*n/a*	*n/a*	*n/a*
	Reserved	E	**240**.0.0.0 - **255**.255.255.255	*n/a*	*n/a*	*n/a*

Figure 5 . IP Classes Comparison.

4. Network Address

The network address is the first address in the network, and it is used for the identification network segment. All the IP addresses, using the same network address part, are in the same network segment. Since the network address is the first address in the network, it cannot be a random IP address, but it must match with network mask in a binary view, for last bits in the network address must be zeros, as long as the mask has zeros. For instance, in the case discussed earlier regarding a given class C IP address, the resulted address of 192.168.10.0 is defined as the network address.

5. Broadcast Address

The broadcast address is the last address assigned in the network, and it is used for addressing all the nodes in the network at the same time. Hence, an IP packet, where the destination address is the broadcast address, is sent to all nodes within the IP network. It is important for remote announcements in the network segment. Further, in some cases, it is used for attacking purposes by hackers or can cause problems in bigger network segments. For instance, in the case discussed earlier regarding a given class C IP address, the resulted address of 192.168.10.255 is defined as the broadcast address.

ii. Cybersecurity

Kaspersky Lab[1], a multinational cybersecurity and anti-virus provider, defines three types of cyber threats:

1. **Cybercrime** includes single actors or groups targeting systems for financial gain or to cause disruption.
2. **Cyber-attack** often involves politically motivated information gathering.
3. **Cyberterrorism** is intended to undermine electronic systems to cause panic or fear.

[1] https://www.kaspersky.com/resource-center/definitions/what-is-cyber-security

Cyber-attacks are usually intended to obtain money, data, or property, or for political purposes, espionage, or disruption. There is a huge number of cyber-attacks occurring worldwide, however, most of them are not published nor reported. In the last years, we are witnessing an increase in the cyber-attacks domain in all kind of sectors: the government sector, private companies, the financial sector, etc. [2].

In most literature, cybersecurity is used as an all-inclusive term. Definitions of this term vary, for example, the *International Telecommunications Union* (ITU) defines cybersecurity as follows: "Cybersecurity is the collection of tools, policies, security concepts, security safeguards, guidelines, risk management approaches, actions, training, best practices, assurance and technologies that can be used to protect the cyber environment, organization, and user's assets. Organization and user's assets include connected computing devices, personnel, infrastructure, applications, services, telecommunications systems, and the totality of transmitted and/or stored information in the cyber environment. Cybersecurity strives to ensure the attainment and maintenance of the security properties of the organization and user's assets against relevant security risks in the cyber environment". Cybersecurity goes beyond the boundaries of traditional information security to include not only the protection of information resources but also that of other assets, including the person him/herself. Such assets include absolutely anyone or anything that can be reached via cyberspace [3].

To threaten cyber-security, malicious actors use some common methods:

- **Malware** – malicious software. One of the most common cyber threats, malware, is a software that a cybercriminal or hacker has created to disrupt or damage a legitimate user's computer. There are several different types of malware, including viruses, worms, trojans, spyware, ransomware, adware, and botnets.
- **SQL injection** – a type of cyber-attack used to take control of and steal data from a database.
- **Phishing** – when cybercriminals target victims with emails that appear to be from a legitimate company asking for sensitive information.
- **Man-in-the-middle attack** – a type of cyber threat where a cybercriminal intercepts communication between two individuals in order to steal data.
- **Denial-of-service attack** – where cybercriminals prevent a computer system from fulfilling legitimate requests by overwhelming the networks and servers with traffic.

iii. Computer Worms

A computer worm is a type of malicious software program whose primary function is to infect other computers while remaining active on infected systems, hence, self-replicating and duplicates itself and spreads through networks [4]. Worms can spread very quickly and disrupt system use by clogging the network. It is common for worms to be noticed only when their uncontrolled replication consumes system resources, slowing or halting other tasks. Once the worm is detected, it is often easy to patch the system and prevent it from spreading further within the network. Also, note the difference between a computer virus and worms in which worms do not require activation or any human intervention to execute or spread their code.

Computer worms often rely on the vulnerabilities in networking protocols to propagate. For instance, the *WannaCry* ransomware[2] worm exploited a vulnerability in the first version of the Server Message Block (SMBv1) resource sharing protocol implemented in the Windows operating system. Once active on a newly infected computer, the WannaCry malware initiates a network search for new potential victims using the broadcast address of the infected computer. Then, the worm can continue to propagate within an organization among systems that respond to SMBv1 requests made by the worm. When a bring your own device (BYOD) is infected, the worm can spread to other networks, giving hackers even more access.

As can be seen in Figure 6, sophisticated worms such as *Conficker* [5] have three main spreading strategies. First, Local Spreading (LS) where the worm in an infected computer probes computers in the same Local Area Network (LAN) with the same IP address prefix. Second, Neighborhood Spreading (NS) where it probes computers in ten neighboring LANs (with smaller consecutive IP address prefixes). Third, Global Spreading (GS) where the worm probes computers with random IP addresses on the Internet. In addition to their spreading strategies, sophisticated worms are prioritized to replicate among computers with higher importance such as Servers e.g. Databases (DB), Storage, etc., as will be elaborated in the next subsection. Furthermore, sophisticated worms can diagnose the security mechanism of a given potential computer victim that affects the order of the worm's infection within the network to increase both infection rate faster and the scale of the computer network's damage.

iv. Servers

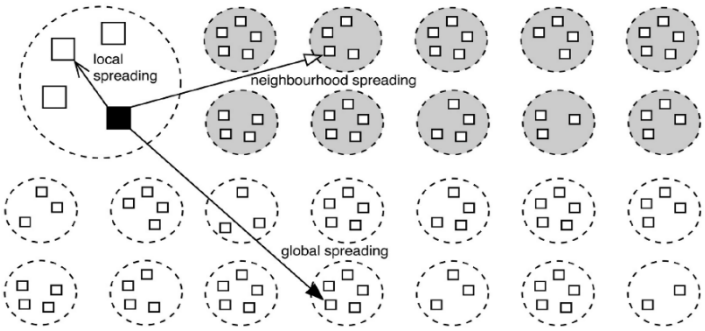

Figure 6. Conficker's three probing strategies (Zhang et al., 2015)

In computing, a server is a computer program or a device that provides functionality for other programs or devices, called "clients". This architecture is called the client-server model. Servers can provide various functionalities, often called "services", such as sharing data or resources among multiple clients or performing computation for a client. A single server can serve multiple clients, and a single client can use multiple servers. A client process may run on the same device or may connect over a network to a server on a different device [3].

Servers are often dedicated, meaning that they perform no other tasks besides their server tasks. Different servers do different jobs, from serving email and video to protecting

[2] WannaCry ransomware - https://www.kaspersky.com/resource-center/threats/ransomware-wannacry

internal networks and hosting Web sites[3]. Table 1 presents some of the most crucial and widely used different types of servers and their purpose.

Server Type	Purpose
Application Server	A server that hosts web apps (applications, computer programs that run inside a web browser) allowing users in the network to run and use them, without having to install a copy on their own computers[4].
Computing Server	A server that shares vast amounts of computing resources, especially CPU and random-access memory, over a network. Also called a supercomputer.
Database Server	A server that maintains and shares any form of database (organized collections of data with predefined properties that may be displayed in a table) over a network.
FTP Server	A server the shares files and folders, storage space to hold files and folders, or both, over a network. FTP stands for File Transfer Protocol which is the protocol that makes it possible to move one or more files securely between computers while providing file security and organization as well as transfer control.
Mail Server	A server that moves and store mail over corporate networks (via LANs and WANs) and across the Internet. Software that transfers electronic mail messages from one computer to another using SMTP (Simple Mail Transfer Protocol)[5].
Proxy Server	A server that acts as an intermediary between a client program (typically a Web browser) and an external server (typically another server on the Web), accepting incoming traffic from the client and sending it to the server. The reasons for doing so include content control and filtering, improving traffic performance, preventing unauthorized network access, or simply routing the traffic over a large and complex network.
Web Server	A server that hosts web pages. A web server is what makes the World Wide Web possible. Each website has one or more web servers. At its core, a Web server serves static content to a Web browser by loading a file from a disk and serving it across the network to a user's Web browser. This entire exchange is mediated by the browser and server talking to each other using HTTP.

Table 1. Server Types.

Problem Definition

We want to find a quick way to prevent the spreading of a computer worm in a network that consists of hosts (e.g. worm's targets) with different roles and with a diversified amount of security components. As security experts, our goal is to find the strategist hosts from which the worm is more likely to spread to a major number of hosts and therefore cause higher damage to the whole network. Also, It's important to emphasize that since each network consists a range of hundreds to tens of millions of hosts nodes where each node is connected to a substantial amount of local nodes, neighborhood nodes, and global nodes, from combinatorial and computational complexity, we deal with a Non-Polynomial (NP) hardness problem. Therefore, in order to address this issue, we propose a combined

[3] https://www.webopedia.com/quick_ref/servers.asp

[4] https://www.theserverside.com/news/1363671/What-is-an-App-Server

[5] https://tools.ietf.org/html/rfc5598

approach using the Influence Maximization Independent Cascade Model [6], Memetic Algorithm [7], and Random Forest Regressor [8].

Methods

i. Data Generation

Network generation

In this phase, we used the *ipaddress* python library for IPv4/IPv6 manipulation. Our HostNode class represents a single node in the network and is used in our DataGeneration class for our network creation. We created two different networks: Network A contains the hosts with IP addresses in the range of 192.168.7.0 - 192.168.7.255. Network B contains the hosts with IP addresses in the range of 10.10.8.0 - 10.10.8.255. The IP addresses that were chosen are used for denoting internal IP addresses for internal communication between the networks' nodes. To create a large unified network, we defined 15 nodes in each network as "connectors". Each connector from subnetwork A is connected to all the nodes within its subnetwork as well as to another connector node from subnetwork B, and vice versa. The network structure will be further explained in the *Networkx Graph* **creation** subsection.

Each host node has the following features:

1. **Network Features** - IP Address, Subnet Mask, Network Address, and Broadcast Address. These features were generated by using the *ipaddress* python library for IPv4/IPv6 manipulation.
2. **Node Features** – Node Type (host/server), Server Type (host/ Application/Computing/Database/FTP/Mail/Proxy/Web), and Importance Score. The features Node Type and Server Type were generated using probabilities that were chosen and estimated based on prior knowledge and research. Figure 7 presents the generation schema for these features and the chosen probabilities for each possibility. Each node has a probability of 0.8 for being generated as a host, and 0.2 for being generated as a server. If a given host node is defined as a host, its server type will also be a host. However, in case the host node is defined as a server, the server type is randomly chosen from a list of server types, while there are different probabilities for each one of them. For instance, since Database and Mail servers are very common and needed in almost every network, the probability of this type of server is higher than the probability of proxy server since there are less frequent.

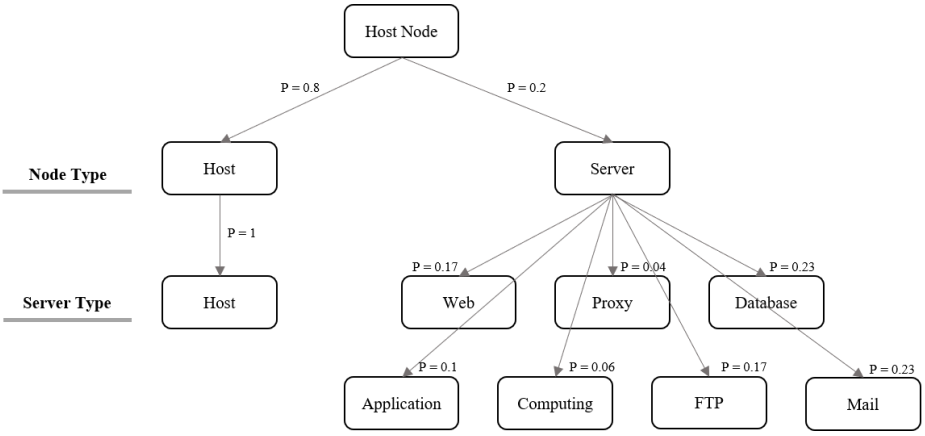

Figure 7. Nodes and Server types generation.

The Importance Score feature is a grade that was also estimated based on prior knowledge and research. This feature's purpose is to score the different nodes and server types by their importance for the company. As mentioned in the previous subsection, sophisticated worms are prioritized to replicate among computers with higher importance such as some of the specified server types. We wanted to simulate this scenario and to give each node an importance score, similarly to the worm's infection method. The importance score given for each node is presented in Table 2.

Node Type	Prior Probability for Node Type	Server Type	Prior Probability for Server Type	Importance Score
Host	0.8	Host	1	1
Server	0.2	Application	0.1	6
		Computing	0.06	5
		Database	0.23	8
		FTP	0.17	7
		Mail	0.23	8
		Proxy	0.04	9
		Web	0.17	6

Table 2. Node's Features.

3. **Security Components Features** – A list of security components, Security Weight, and Spread Weight. There are many existing security components. We chose to produce a list of components, policies, and mechanisms that are the most used for securing a network. It should be noted that some of these components were not created for this purpose originally, but nowadays are also used for security. For instance, a proxy server is a server application or appliance that acts as an intermediary for requests from clients seeking resources from servers that provide those resources [9]. The proxy server can control these request, but also provides additional benefits such as load balancing, privacy, or security and are highly recommended for use in terms of cybersecurity [10]. The security components we chose are Firewall, OS Version, Antivirus, Security Policies, VPN, and Proxy.
Similarly to the generation of the nodes' features, we also generated the security components that each node has. In this case, we first randomly generated for each node the number of security components, and later we generated the components types.
The security components and their probabilities are presented in Table 3 below. Each of these security components was given a Security Weight based on its known security capabilities in a network. This information is also presented in Table 3.

Security Component	Prior Probability	Security Weight (W)
Firewall	0.3	0.3
OS Version	0.1	0.15
Antivirus	0.3	0.2
Security Policies	0.15	0.15
VPN	0.05	0.1
Proxy	0.1	0.1

Table 3. Security components and their generation probabilities.

As can be seen, the security weights were normalized, and their sum equals 1. In our problem settings, each node can have each component only once, i.e. each node can either be connected to a Firewall, or not, but there is no meaning of being connected to more than one Firewall. The Security Weight feature of a single node is the sum of all security weights of the security components it has, as presented in Equation 1.

$$Security\ Weight_{node\_i} = \sum_{\substack{security \\ component=1}}^{n} W_{security\ component} \quad (1)$$

The Spread Weight is calculated using Equation 2 and represents the spreading weight of the worm.

$$Spread\ Weight_{node\_i} = 1 - \ Total\ Security\ Weight_{node_i} \ (2)$$

In that way, if a node has all the security components – it is completely secured. Its total security weight will be 1 and its spread weight will be 0 – meaning the worm will not be capable of infecting it. This design is based on the idea that theoretically, a node can be nearly completely secured if all existing resources will be invested in its security, but it is not cost-effective to do so, and the responsible security experts are always trying to find the most cost-effective way to secure their network.

4. **Connectors Features** – Is Connector, Connected Host, Connected Hosts Amount.
   As mentioned, some of the nodes were defined as connectors between the two subnetworks which form a larger unified network. Each node has a Boolean feature named Is Connector. TRUE values are given to 30 nodes – 15 from each of the subnetworks. For each of the connector hosts, the feature Connected Host contains the IP address of the host in the other network that the node is connected to. Connected Hosts Amount contains the number of nodes the node is connected to. In most cases, the value is 254, and if the node is a connector – the value turns to 255.
5. **Node's Total Score Feature** – In the *Computer Worms* section, we described the spreading technique of a sophisticated worm and we mentioned two major issues that affect it: the importance of the computers in the network (the higher the importance is – the higher the computer is prioritized), and the spread weight each computer that is based on the security mechanism of a given potential computer victim (the worm will prioritize a node with a high spread weight, which means a low-security level). Based on these ideas, we created a feature that represents the total score given to each node in the network, from the worm's perspective. this feature is calculated as presented in Equation 3. To create a total weighted score for each node we normalize the importance score of each node (each importance score was divided to the sum of the existing importance scores). We chose to give a higher weight of 0.6 to the node's normalized importance score and a lower weight to its spread weight.

$$Total\ Score_{node_i} = 0.4 \times Spread\ Weight_{node_i} + 0.6 \times Normalized\ Importance\ Score_{node_i} \ (3)$$

Networkx Graph creation

We used the created data frame described above to create a weighted directed graph. To do so, we used DiGraph of *networkx* library. the graph is composed of nodes and weighted directed edges that were created as follows: we first converted each row in our data frame that represents a hostNode object into a node in the graph. Then, we created two edges between each couple of connected nodes in both two possible directions (from node A to node B, and vice versa). The nodes connected with an edge are all the nodes in network A that are connected all one to another; All the nodes in subnetwork B; and the nodes that were defined as connectors between the two subnetworks are connected – each node is connected to another node in the other subnetwork. Figure 8 shows the network structure from a bird's eye view. Even though not all the nodes and edges are readable, one can still understand from it the basic structure of two subnetworks A and B (encircled with green circles) that are connected and combined to form a unified larger network using some pre-defined connectors nodes (encircled with red circles). The specified weight of each edge

denoted as $Total\ Score_{node_i}$, is defined as the total score of the node that the edge is directed to, which is presented in Equation 3. We chose to define the weights in this way since the total score represents the ability of the worm to prioritize, penetrate and infect a given node, and therefore is affected by the score of the node into which the edge enters.

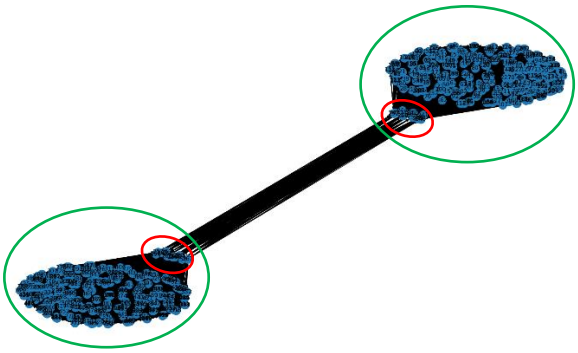

Figure 8. The network's structure.

ii. Influence Maximization – Independent Cascades

Social Network is increasing its popularity, and one of the important research areas within this field is Information Diffusion, which is the process by which a new idea spread over the networks based on communication among the social entities. The problem of finding a seed set composed of k nodes that maximize their influence spread over a social network called Influence Maximization (IM). Further, two widely used diffusion models are *the Linear Threshold Model* (LTM) and the *Independent Cascade Model* (ICM). In this article, we will focus on the ICM due to its relevancy to our problem defined earlier.

Kempe et al. [6] proofed that the influence maximization problem is NP-hard for the ICM problem using a reduction based on the *Set Cover* problem, which aims to find the smallest sub-collection of S whose union equals the universe, where universe stands for a set of elements $\{1,2,\ldots,n\}$, and S defined as a collection of m sets whose union also equals the universe itself. Kempe et al. also proposed a greedy algorithm that guarantees a 63% influence spread of the optimal solution.

To better understand how ICM works, consider a point in the cascade process when a node v has just become active, and therefore, attempts to activate its neighbor node w with a probability of success $p_{v,w}$. The outcome of such a process is similar to a random event that its outcome is being determined by flipping a coin with a bias $p_{v,w}$. Note that it does not matter whether the coin was flipped at the moment that v became active, or flipped at the very beginning of the whole process and is only being revealed now. As a result, we can assume that for each pair of neighbors (v,w) in the network graph, a coin of bias $p_{v,w}$ is flipped at the very beginning of the cascade process independently of the coins belong to the rest of the neighbors' pairs, and the result is stored so that it can be later checked when v is activated while w remains inactive. With all the coins flipped in advance, the cascade process can be viewed as follows. The edges in graph G for which the coin flip indicated an activation will be successful are declared to be live, hence, the remaining edges are declared as blocked. The ICM process runs in discrete steps, where at the beginning, few nodes are given the information known as seed nodes. Upon receiving the information these nodes become active. In each discrete step, an active node tries to influence one of its inactive neighbors. also, note that the same node will never get another chance to activate the same inactive neighbor. The success depends on the propagation probability of their

connection, e.g. undirected\directed edge. An edge propagation probability is a probability by which one can influence the other node. In practice, the propagation probability is relation dependent where each edge will have a different value. Finally, the ICM process terminates when no further nodes became activated from an inactive state. Furthermore, since the ICM is a stochastic process, it requires a diffusion simulation to be executed for a sufficiently large number of times to accurately determine the expectation of the information diffusion spread.

A demonstration of a single diffusion process based on ICM taken from the book of Chen et al. [11] is demonstrated in Figure 9 , where orange nodes denote active nodes, and green nodes denote inactive nodes. Solid green edges represent originally directed edges within the network graph and the green numbers near them represent the influence probabilities. As can be seen in time step 1, the solid red edge from node v_1 to node v_5 means that v_1 successfully activates v_5 through this edge. However, the dotted green edge from node v_1 to a node v_3 means that v_1 fails to activate v_3 through this edge.

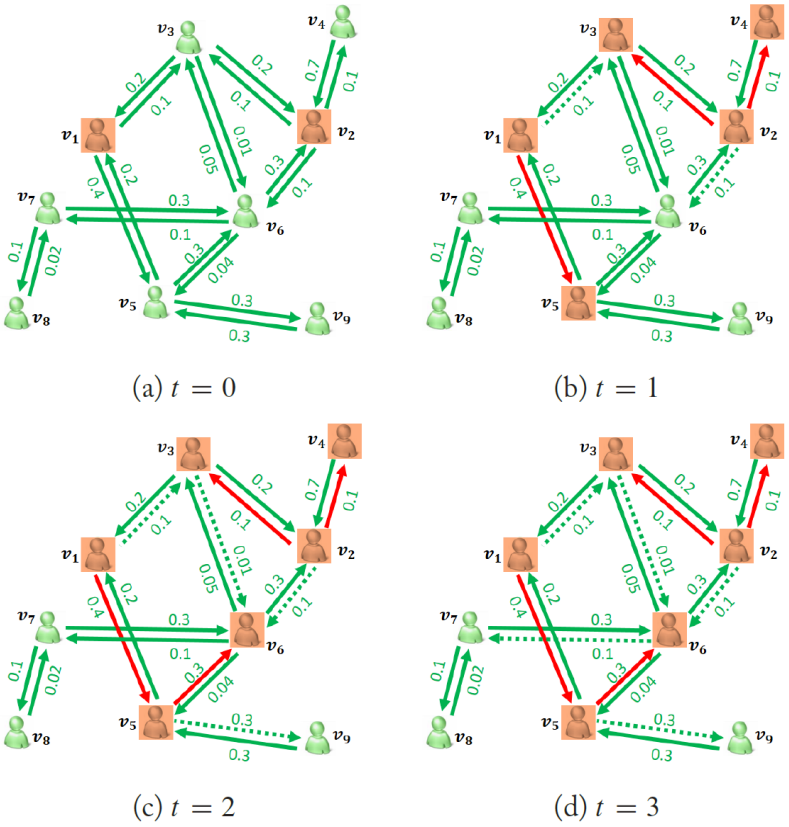

Figure 9 . ICM Diffusion process (Chen et al, 2013)

iii. Memetic Algorithm

A common branch of evolutionary computation, known as the Memetic Algorithm (MA) [7], is a hybrid of global search methods and local search procedures. Global search methods are usually based on evolutionary and swarm intelligence [12], and as a result, can produce a reliable estimate of the global optimum. Local search procedures though are individual refinement processes that involve domain-knowledge, and therefore, can explore for better solutions around the best solution found so far. MAs have been proved to play an important role in solving complex optimization problems in social networks.

To address the problem of finding the ultimate set of the most influential nodes, denoted as seed, Gong et al. [13] proposed a novel memetic algorithm for influence maximization in social networks, which was implemented and modified to our scenario and goals. This algorithm combines a genetic algorithm as the global search method and a similarity-based and nearest-neighbor strategy as the local search procedure. As can be seen in the Meme-IM algorithm presented in Figure 10 , in **Step 1)**, the ***populationInitialization*** method, mainly completes the population initialization task. First, it creates a population of solutions, referred to as an individual chromosomes $P = \{x_1, x_2, \ldots, x_{popSize}\}$, in which each chromosome represented by a set of host nodes that serves as the initial origin seed for the influence maximization algorithm.

The population is initialized as follows; Initially, half of the existed host nodes, i.e. genes, are being selected using the *Similarity-based High Degree method* (SHD). For each host node candidate, SHD first locates the host node with the highest out-degree, i.e. the node with the maximal edges pointed out to other host nodes within the network. That is, in the sense that a bigger out-degree is more likely to influence the whole network. Then, SHD adds the node with the highest out-degree to the initial population set and excludes it from the set of candidates nodes. In order to rich and select a heterogeneous and diversified population, the next step for SHD is to update the candidate set by excluding nodes similar to the last chosen node using a similarity threshold denoted as *simThreshold*. In this paper, due to the use of IPs as the unique identifier of a host's node and since similar prefix for two IPs means, that they are within the same community i.e. the same subnetwork, we chose to use *Jaccard Similarity* (JS) measure calculated using the following Equation 4;

$$Jaccard\ Similarity(IP_A, IP_B) = \frac{|\ IP_A \cap IP_B\ |}{|\ IP_A \cup IP_B\ |} \quad (4)$$

For instance, given two IPs: IP_A, IP_B with addresses of $192.168.110.026, 192.168.120.016$ respectively, it can be seen that their digit-wise intersection is equal to 10 out of 12. Hence, their JS score equal to 83.33%, which means that they are highly similar.

After retrieving a set of host nodes candidates that represents half of the population using the SHD algorithm, chromosomes with a size of k (i.e. seedSize) are created by uniformly at random select k host nodes for each chromosome taken from the candidates. Last, for the other half of the chromosomes population, a chromosome with a size of k is generated also using the SHD algorithm, however, for each gene (i.e. host node) in the chromosome, there's a 50 percent chance of replacing the gene with a randomly selected gene within the population (network).

After the initial population generated using the abovementioned procedure, using Equation 5 the Meme-IM algorithm selects the chromosome with the maximum fitness as P_{best}, i.e. the seed of k host nodes with the highest IM-Independent Cascade Model Ratio;

$$IM - ICM\ Ratio\colon Fitness(seed) = \frac{Total\ Infected\ Host\ Nodes\ (seed)}{min(maxIterations, Total\ Iteration\ till\ convergence\ (seed))} \quad (5)$$

Note that the purpose of choosing the fitness as the ratio and not only the total infected hosts is to normalize it by the number of iterations took until convergence, hence, adding the time factor. For instance, if two different seeds resulted in the same amount of infected hosts, however, the total infected hosts influenced by the 1st seed converged after 10 iterations while the 2nd seed configuration converged after 15 iterations. As a result, we

can compare both seeds and infer that the 1st seed has much more influence compared to other seeds' influence. Therefore, we thought that this heuristic is much more informative compared to the fitness of just observing the total infected hosts.

Memetic Influence Maximization Algorithm (Meme-IM)

Input: Network connection matrix: *G*, Maximum generation: *maxGen*,
population size: *popSize*, seed size: *seedSize,* mating pool size: *parentsPoolSize*,
tournament size: *tourSize*, crossover probability: *crossProb*,
and mutation probability: *mutateProb*.
Output: The most influential *k*-node set.
1: **Step 1) Initialization**
2: **Step 1.1)** Population initialization:
$P = \{x_1, x_2, \ldots, x_{popSize}\}$;
3: **Step 1.2)** Best individual initialization: $P_{best} = x_i$;
4: **Step 2)** Set $t = 0$; // the number of generations
5: **Step 3) Repeat**
6: **Step 3.1)** Select parental chromosomes for mating:
$P_{parent} \leftarrow \boldsymbol{parentsSelection}(P, parentsPoolSize, tourSize)$;
7: **Step 3.2)** Perform genetic operators:
$P_{child} \leftarrow \boldsymbol{geneticOperation}(P_{parent}, crossProb, mutateProb)$;
8: **Step 3.3)** Perform local search:
$P_{new} \leftarrow \boldsymbol{localSearch}(P_{child})$;
9: **Step 3.4)** Update population:
$P \leftarrow \boldsymbol{updatePopulation}(P, P_{new})$;
10: **Step 3.5)** Update the best individual P_{best};
11: **Step 4) Stopping criterion**: If $t < maxGen$, then
$t = t + 1$ and go to **Step 3**), otherwise, stop the
algorithm and output.

Figure 10 . Meme-IM algorithm

Step 3) is the evolution procedure. In **Step 3.1)**, based on the given $tourSize$, and the required parents' pool size ($parentsPoolSize$), Meme-IM first uses a deterministic tournament selection method to select parental individuals $Pparent$ for mating in genetic algorithm. In each tournament, the first chromosome to be selected for the parents' pool, competes against another randomly selected chromosomes until the best one to be found within a whole tournament process is selected. This procedure repeats itself until achieving the required pool size. Then in **Step 3.2)**, Meme-IM reproduces the chosen parental individuals *Pchild* using the $\boldsymbol{geneticOperation}$ procedure, i.e., performs $\boldsymbol{crossover}$ and $\boldsymbol{mutation}$ operation on *Pparent*. In our scenario, crossover works as follows; first, for every parent chromosome, there's a chance of $crossProb$ to be selected for the crossover process. Then, for every two chromosomes' selected parents with no similar genes, i.e. similar host nodes, a one-point crossover position between a range of $[0, k]$ is randomly selected to create two offsprings generated based on the random position. For instance, in the case of two parents' chromosomes of size 3, and a random position of 2, Offspring A will be consist of the first two genes (host nodes) within Chromosome A and the last gene of Chromosome B, whereas Offspring B, will be consist of the first two genes of Chromosome B and the last gene of Chromosome A. After crossover, the resulted generation, $Pchild$, goes to a mutation procedure in which for each gene in every child chromosome, there's a chance of $mutateProb$ to be muted. Hence, each gene can be replaced by a random gene that isn't similar (based on JS score) to the existed genes within the chromosome.

Step 3.3) aims to find a better solution (i.e. chromosome) around the best chromosomes found so far, a Local search procedure ($\boldsymbol{localSearch}$) is being performed by employing a nearest neighbor-based strategy. This procedure is performed on the best individual chromosome retrieved from a certain generation, and then it attempts to find a better chromosome from the neighborhoods of the best individual in $Pchild$. The fitness is

calculated using Eq. 2 and the neighbor individual is better if its fitness is larger than that of the original individual. Hence, this change is accepted if a better individual can be located. The procedure repeats until no further improvement can be made. Finally, **Step 3.4)**, is to refresh the current population by taking the best individuals from $P \cup Pnew$. And in **Step 4)**, when the algorithm terminates on convergence, Meme-IM stops and outputs the ultimate k node-set.

iv. Random Forest Regressor

Random forests (or Random Decision Forests) are an ensemble learning method for classification, regression, and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees [8].

We decided to use this method for two main goals:

1. **Prediction -** Prediction of the probability of a given specific node to get infected. This is the last phase of our system. The output of the Memetic Algorithm contains a list of origin nodes that we want to estimate their infection probability.
2. **Feature Importance Extraction** – As mentioned in the *Problem Definition* section, our goal is to find the strategist hosts from which the worm is more likely to spread to a major number of hosts and therefore cause higher damage to the whole network. to do so and to create an understandable solution, it is important to understand which are the most significant features that affect the probability for a node to be infected.

The input for the Random Forest Regressor is a data frame based on the network data that was presented in the *Data Generation* section, with some modifications including features selection, creation of new relevant features, and oversampling technique.

The features from the network data that are selected are: Node Type, Server Type, Importance Score, Is Connector, Connected Hosts Amount.

The feature of Security Components that contains for each node a list of its security components, was transformed using a one-hot encoding representation. We created the features: Firewall, OS Version, Antivirus, Security Policies, VPN, and Proxy. For each node, if the node has a particular security component – the feature gets the value of "1", and else "0".

New features were created in the Independent Cascades Model's (ICM) iterations. The new feature "Nodes infected in 1 iter" represents the number of nodes that were infected in the first iteration when conducting the ICM when each node is defined as the origin of each ICM execution. This feature is based on the idea that if the number of nodes infected by this origin-node, it may apply that this node is significant for the worm spreading in the network.

Our target (Y) variable is defined in Equation 6 as:

$$Y_{node_i} = P(infection\ of\ node_i) = \frac{number\ of\ times\ in\ which\ node_i\ was\ infected}{number\ of\ infections\ opportunities\ for\ node_i} \quad (6)$$

The numerator in this fraction, the number of times node i was infected, is also calculated during the ICM procedure. The ICM was executed several times which is equal to the number of nodes in the network (in our case – 508). At each of these episodes, the node-origin was one of the existed nodes within the network. Each time a node in the network was infected, its number of infections increased in 1.

The denominator in this fraction, the number of infections opportunities for node i, is the number of executions of the ICM model which is in our case – 508.

To create a reasonable amount of data as an input to the regressor model, and to calculate the expectation of each node to be infected, we oversampled the data. We executed the ICM model from each node-origin three times, each time using a different random seed.

Features importance

The features importance plot that was extracted from our trained Random Forest Regressor is presented in Figure 11. As one can see, the most significant features are the security components: VPN, Proxy, OS Version, and Security Policies, by this order. The order of the different security components is compatible with our initial definition of security weight that was given for each component. The higher the security weight of a component is, the lower is the feature's importance. It is important to emphasize that even though we didn't include the security weight values in the training set, there is a correspondence between the real security components weights to their columns which appear in the feature importance plot. Hence, we can confirm that our trained RF regressor learned a meaningful set of hypotheses. The number of nodes infected in the 1st iteration, the number of hosts connected directly to the node, and the Boolean Is Connector feature are all features that are practically not important at all.

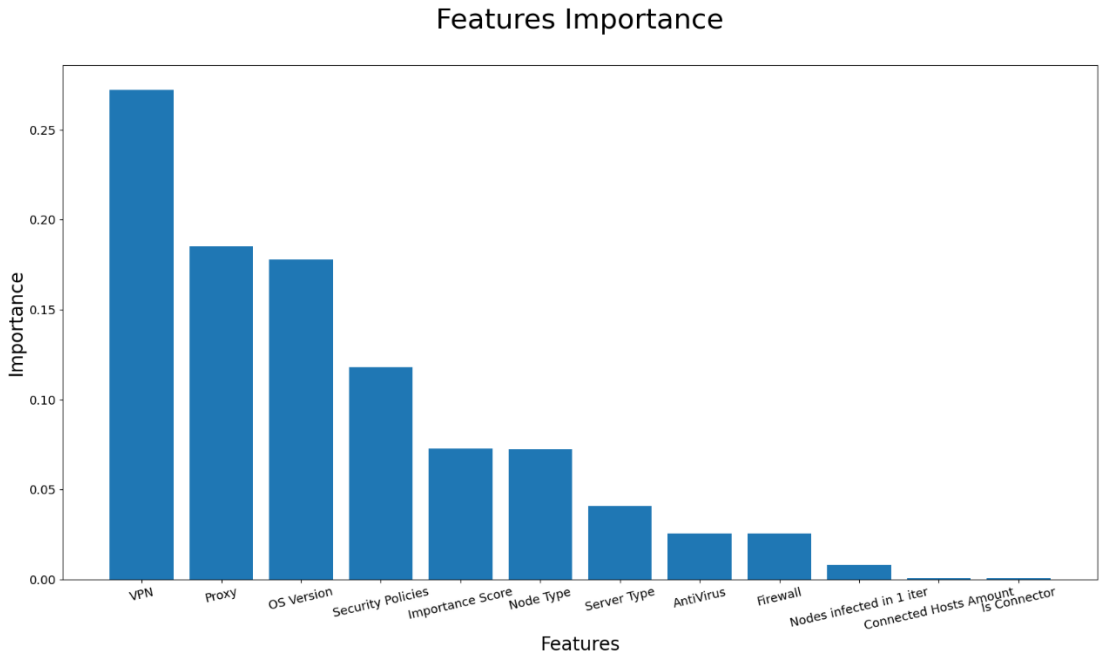

Figure 11. Features Importance

Hyperparameters tuning

To find which regressor's parameters will gain the best results, we tunned the hyperparameters of the model. We executed this phase using K-folds Cross-Validation (K=5). We tried 144 different combinations of the model parameters values. The results of this phase are presented in Table 4, and the best model that was chosen for our regressor and the values of its parameters are colored red.

Model index	N estimators	Max depth	Criterion	Min samples split	Bootstrap	MSE	Model index	N estimators	Max depth	Criterion	Min samples split	Bootstrap	MSE
1	200	10	MSE	2	TRUE	9.92E-06	73	1000	10	MSE	2	TRUE	1.54E-05
2	200	10	MSE	2	FALSE	1.06E-05	74	1000	10	MSE	2	FALSE	1.54E-05
3	200	10	MSE	5	TRUE	1.04E-05	75	1000	10	MSE	5	TRUE	1.53E-05
4	200	10	MSE	5	FALSE	1.05E-05	76	1000	10	MSE	5	FALSE	1.52E-05
5	200	10	MSE	10	TRUE	1.04E-05	77	1000	10	MSE	10	TRUE	1.52E-05
6	200	10	MSE	10	FALSE	1.05E-05	78	1000	10	MSE	10	FALSE	1.51E-05
7	200	10	MAE	2	TRUE	1.42E-05	79	1000	10	MAE	2	TRUE	1.54E-05
8	200	10	MAE	2	FALSE	1.42E-05	80	1000	10	MAE	2	FALSE	1.53E-05
9	200	10	MAE	5	TRUE	1.65E-05	81	1000	10	MAE	5	TRUE	1.56E-05
10	200	10	MAE	5	FALSE	1.62E-05	82	1000	10	MAE	5	FALSE	1.56E-05
11	200	10	MAE	10	TRUE	1.82E-05	83	1000	10	MAE	10	TRUE	1.58E-05
12	200	10	MAE	10	FALSE	1.79E-05	84	1000	10	MAE	10	FALSE	1.58E-05
13	200	50	MSE	2	TRUE	1.75E-05	85	1000	50	MSE	2	TRUE	1.58E-05
14	200	50	MSE	2	FALSE	1.72E-05	86	1000	50	MSE	2	FALSE	1.58E-05
15	200	50	MSE	5	TRUE	1.68E-05	87	1000	50	MSE	5	TRUE	1.57E-05
16	200	50	MSE	5	FALSE	1.66E-05	88	1000	50	MSE	5	FALSE	1.57E-05
17	200	50	MSE	10	TRUE	1.63E-05	89	1000	50	MSE	10	TRUE	1.56E-05
18	200	50	MSE	10	FALSE	1.61E-05	90	1000	50	MSE	10	FALSE	1.56E-05
19	200	50	MAE	2	TRUE	1.61E-05	91	1000	50	MAE	2	TRUE	1.56E-05
20	200	50	MAE	2	FALSE	1.61E-05	92	1000	50	MAE	2	FALSE	1.56E-05
21	200	50	MAE	5	TRUE	1.61E-05	93	1000	50	MAE	5	TRUE	1.56E-05
22	200	50	MAE	5	FALSE	1.61E-05	94	1000	50	MAE	5	FALSE	1.56E-05
23	200	50	MAE	10	TRUE	1.62E-05	95	1000	50	MAE	10	TRUE	1.56E-05
24	200	50	MAE	10	FALSE	1.61E-05	96	1000	50	MAE	10	FALSE	1.56E-05
25	200	100	MSE	2	TRUE	1.60E-05	97	1000	100	MSE	2	TRUE	1.56E-05
26	200	100	MSE	2	FALSE	1.59E-05	98	1000	100	MSE	2	FALSE	1.56E-05
27	200	100	MSE	5	TRUE	1.57E-05	99	1000	100	MSE	5	TRUE	1.55E-05
28	200	100	MSE	5	FALSE	1.56E-05	100	1000	100	MSE	5	FALSE	1.55E-05
29	200	100	MSE	10	TRUE	1.55E-05	101	1000	100	MSE	10	TRUE	1.55E-05
30	200	100	MSE	10	FALSE	1.54E-05	102	1000	100	MSE	10	FALSE	1.54E-05
31	200	100	MAE	2	TRUE	1.54E-05	103	1000	100	MAE	2	TRUE	1.54E-05
32	200	100	MAE	2	FALSE	1.54E-05	104	1000	100	MAE	2	FALSE	1.54E-05
33	200	100	MAE	5	TRUE	1.55E-05	105	1000	100	MAE	5	TRUE	1.55E-05
34	200	100	MAE	5	FALSE	1.55E-05	106	1000	100	MAE	5	FALSE	1.54E-05
35	200	100	MAE	10	TRUE	1.56E-05	107	1000	100	MAE	10	TRUE	1.55E-05
36	200	100	MAE	10	FALSE	1.56E-05	108	1000	100	MAE	10	FALSE	1.55E-05
37	500	10	MSE	2	TRUE	1.54E-05	109	2000	10	MSE	2	TRUE	1.54E-05
38	500	10	MSE	2	FALSE	1.53E-05	110	2000	10	MSE	2	FALSE	1.54E-05
39	500	10	MSE	5	TRUE	1.52E-05	111	2000	10	MSE	5	TRUE	1.53E-05
40	500	10	MSE	5	FALSE	1.51E-05	112	2000	10	MSE	5	FALSE	1.53E-05
41	500	10	MSE	10	TRUE	1.49E-05	113	2000	10	MSE	10	TRUE	1.52E-05
42	500	10	MSE	10	FALSE	1.48E-05	114	2000	10	MSE	10	FALSE	1.52E-05
43	500	10	MAE	2	TRUE	1.53E-05	115	2000	10	MAE	2	TRUE	1.54E-05
44	500	10	MAE	2	FALSE	1.53E-05	116	2000	10	MAE	2	FALSE	1.54E-05
45	500	10	MAE	5	TRUE	1.57E-05	117	2000	10	MAE	5	TRUE	1.55E-05
46	500	10	MAE	5	FALSE	1.57E-05	118	2000	10	MAE	5	FALSE	1.55E-05
47	500	10	MAE	10	TRUE	1.61E-05	119	2000	10	MAE	10	TRUE	1.57E-05
48	500	10	MAE	10	FALSE	1.61E-05	120	2000	10	MAE	10	FALSE	1.57E-05
49	500	50	MSE	2	TRUE	1.60E-05	121	2000	50	MSE	2	TRUE	1.57E-05
50	500	50	MSE	2	FALSE	1.60E-05	122	2000	50	MSE	2	FALSE	1.57E-05
51	500	50	MSE	5	TRUE	1.59E-05	123	2000	50	MSE	5	TRUE	1.56E-05
52	500	50	MSE	5	FALSE	1.58E-05	124	2000	50	MSE	5	FALSE	1.56E-05
53	500	50	MSE	10	TRUE	1.58E-05	125	2000	50	MSE	10	TRUE	1.56E-05
54	500	50	MSE	10	FALSE	1.57E-05	126	2000	50	MSE	10	FALSE	1.55E-05
55	500	50	MAE	2	TRUE	1.57E-05	127	2000	50	MAE	2	TRUE	1.55E-05
56	500	50	MAE	2	FALSE	1.57E-05	128	2000	50	MAE	2	FALSE	1.55E-05
57	500	50	MAE	5	TRUE	1.57E-05	129	2000	50	MAE	5	TRUE	1.56E-05
58	500	50	MAE	5	FALSE	1.57E-05	130	2000	50	MAE	5	FALSE	1.55E-05
59	500	50	MAE	10	TRUE	1.58E-05	131	2000	50	MAE	10	TRUE	1.56E-05
60	500	50	MAE	10	FALSE	1.57E-05	132	2000	50	MAE	10	FALSE	1.56E-05
61	500	100	MSE	2	TRUE	1.57E-05	133	2000	100	MSE	2	TRUE	1.55E-05
62	500	100	MSE	2	FALSE	1.57E-05	134	2000	100	MSE	2	FALSE	1.55E-05
63	500	100	MSE	5	TRUE	1.56E-05	135	2000	100	MSE	5	TRUE	1.55E-05
64	500	100	MSE	5	FALSE	1.55E-05	136	2000	100	MSE	5	FALSE	1.55E-05
65	500	100	MSE	10	TRUE	1.55E-05	137	2000	100	MSE	10	TRUE	1.55E-05
66	500	100	MSE	10	FALSE	1.54E-05	138	2000	100	MSE	10	FALSE	1.54E-05
67	500	100	MAE	2	TRUE	1.55E-05	139	2000	100	MAE	2	TRUE	1.54E-05
68	500	100	MAE	2	FALSE	1.54E-05	140	2000	100	MAE	2	FALSE	1.54E-05
69	500	100	MAE	5	TRUE	1.55E-05	141	2000	100	MAE	5	TRUE	1.54E-05
70	500	100	MAE	5	FALSE	1.55E-05	142	2000	100	MAE	5	FALSE	1.54E-05
71	500	100	MAE	10	TRUE	1.55E-05	143	2000	100	MAE	10	TRUE	1.55E-05
72	500	100	MAE	10	FALSE	1.55E-05	144	2000	100	MAE	10	FALSE	1.55E-05

Table 4. Hyperparameters Tuning.

Model's results

We chose to evaluate our model with the most common measure for regression models, which is the Mean Squared Error measure (MSE). The MSE is a measure of the quality of an estimator that measures the average of the squares of the errors - that is, the average squared difference between the estimated values and the actual value. The MSE is always non-negative, and values closer to zero are better.

The MSE result of the chosen model: 8.299826935959292e-06, a value that is very close to zero.

```
MSE results is:  8.299826935959292e-06
```

Results

Before executing the implemented Meme-IM algorithm on our generated network, we configured the hyperparameters with the following settings; First, we chose an initial population size of 40 candidates host nodes. Then, in order to evaluate the ICM ratio of each one of the candidates in a reasonable time, we set up the maximal ICM iteration to 20. Second, in terms of genetics operations, in the parents' selection step, we used a parent's pool size of 18, with a tournament size of 5. Further, the maximal size of generations to be created was set to 3 and for each parent, the probability of being crossover was set to 0.5 since we wanted it to be equally likely. In the aspect of mutation operation, the mutation probability was set to 0.3 since we didn't want to bias our similarity heuristics results which can lead to a slower convergence to the optimal solution space. Furthermore, we used a similarity threshold of 0.6 in each one of our implemented similarity-based methods such as SHD and nearest neighbors. Finally, after executing the algorithm on our generated network, the resulted seed of the best influential host nodes was retrieved after 13 minutes with an ICM ratio of 101.4. Also, note that our Random Forest predictor estimated the resulted influential seed with a 99.8% chance of being infected. The retrieved seed is consists of the following host nodes:

'**Host 77**'
- IP: 192.168.7.75
- Subnet Mask: 255.255.255.0
- Network Address: 192.168.7.0
- Broadcast Address: 192.168.7.255
- Connected Hosts: 254
- Is Connector: False
- Importance Score: 8
- Security Level: 0.4
- Security Components: ['Proxy', 'Firewall']

'**Host 265**'
- IP: 10.10.8.9
- Subnet Mask: 255.255.255.0
- Network Address: 10.10.8.0
- Broadcast Address: 10.10.8.255
- Connected Hosts: 255
- Is Connector: True
- Importance Score: 1
- Security Level: 0.6000000000000001

- Security Components: ['Firewall', 'Proxy', 'AntiVirus']

'**Host 216**'
- IP: 192.168.7.214
- Subnet Mask: 255.255.255.0
- Network Address: 192.168.7.0
- Broadcast Address: 192.168.7.255
- Connected Hosts: 254
- Is Connector: False
- Importance Score: 1
- Security Level: 0.0
- Security Components: []

By exploring the resulted host nodes we can notice some interesting facts. First, Host 77, located in subnetwork A, has an importance score of 0.8 which makes it a much more attractive target for the computer to be spread into. Also, note that it has a pretty low-security level, hence, a higher spread wight. In the case of Host 265, which belongs to subnetwork B, it has 3 different security components and the lowest importance score. However, despite its security mechanisms, this host is defined as a connector meaning it has access to both subnetworks and therefore can become an intermediate host for the worm to be spread from one subnetwork to another. Last, Host 216, also belongs to subnetwork A and has no security components at all, meaning that the worm can easily penetrate and propagate to it and therefore, can spread faster within its network.

After exploring the results it seems that our solution was successfully able to locate strategist hosts that in case of infection can substantially accelerate the spreading rate of a sophisticated worm which can lead to higher damage to the whole computer network and the organization itself. Further, our trained Random Forest predictor was able to learn important features that correspond to the achieved results and also can serve as a prioritization tool for IT and cybersecurity experts in order to deal more efficiently and effectively with such cyber threats, hence, to straighten their network security and to make it more robust.

References

[1] L. Ortiz, “Computer Networks 5th Edition by Andrew S. Tanenbaum David J. Wetherall,” *Comput. Networks 5th Ed. by Andrew S. Tanenbaum David J. Wetherall* .

[2] G. Martin, P. Martin, C. Hankin, A. Darzi, and J. Kinross, “Cybersecurity and healthcare: How safe are we?,” *BMJ*, vol. 358, pp. 4–7, 2017, doi: 10.1136/bmj.j3179.

[3] R. Von Solms and J. Van Niekerk, “From information security to cyber security,” *Comput. Secur.*, vol. 38, pp. 97–102, 2013, doi: 10.1016/j.cose.2013.04.004.

[4] O. Or-Meir, N. Nissim, Y. Elovici, and L. Rokach, “Dynamic malware analysis in the modern era—A state of the art survey,” *ACM Comput. Surv.*, vol. 52, no. 5, Sep. 2019, doi: 10.1145/3329786.

[5] C. Zhang, S. Zhou, and B. M. Chain, “Hybrid Epidemics-A Case Study on Computer Worm Conficker,” 2015, doi: 10.1371/journal.pone.0127478.

[6] D. Kempe, J. Kleinberg, and É. Tardos, “Maximizing the spread of influence through a social network,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 137–146, doi:

10.1145/956750.956769.

[7] P. Moscato, “On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts-Towards Memetic Algorithms On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts Towards Memetic Algorithms,” 2000.

[8] T. K. Ho, “Random decision forests,” in *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 1995, vol. 1, pp. 278–282, doi: 10.1109/ICDAR.1995.598994.

[9] A. Luotonen and K. Altis, “World-Wide Web proxies,” *Comput. Networks ISDN Syst.*, vol. 27, no. 2, pp. 147–154, 1994, doi: 10.1016/0169-7552(94)90128-7.

[10] M. S. Structure, D. Systems, and P. Int, “Structure and Encapsulation in Distributed Systems : the Proxy Principle Marc Shapiro To cite this version : HAL Id : inria-00444651 Structure and Encapsulation in Distributed Systems : the Proxy Principle,” 2010.

[11] W. Chen, L. V. S. Lakshmanan, and C. Castillo, “Information and Influence Propagation in Social Networks,” *Synth. Lect. Data Manag.*, vol. 5, no. 4, pp. 1–177, Oct. 2013, doi: 10.2200/s00527ed1v01y201308dtm037.

[12] X. Chen, Y. S. Ong, M. H. Lim, and K. C. Tan, “A multi-facet survey on memetic computation,” *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5. pp. 591–607, Oct. 2011, doi: 10.1109/TEVC.2011.2132725.

[13] M. Gong, C. Song, C. Duan, L. Ma, and B. Shen, “An Efficient Memetic Algorithm for Influence Maximization in Social Networks,” *IEEE Comput. Intell. Mag.*, vol. 11, no. 3, pp. 22–33, Aug. 2016, doi: 10.1109/MCI.2016.2572538.