Predict the propagation speed of network worm

Gabriel Cao
Sorbonne Universités
cao.gabriel.367@gmail.com

ABSTRACT

Network worms are malicious programs that replicate itself and propagate through the network, they are one of the main threats in the cybersecurity world. Many techniques such as ACLs, packet filtering, Nullroute are used to mitigate their effects. However, the influence of network structure on the propagation speed of network worms has not been studied yet. Here we provide a clear understanding of how the network structure influence the propagation speed of network worms. By simulating networks using existing graph models, we provide the evolution of propagation speed according to the parameter of chosen graph models. Results demonstrate that the speed variation doesn't follow a linear law when the number of connection in the network decrease, it follows an exponential law. Through a simple model of networks with graph theory we could achieve a first simulation of propagation speed of network worms. Future work may use a more complex and realistic model of networks with graph theory to improve the quality of the simulation leading to more accurate results.

1 INTRODUCTION

A lot of computer worms exists and harm our devices. Several techniques are used to prevent the infection from computer worm however when zero-day vulnerabilities are discovered, computer worm use it to get into our device system and spread through the network. The spreading of network worms is a real problem, analog to real virus if a virus doesn't spread quickly enough, it can be killed really quickly and no real problem arise from this, but when a virus spread quickly it becomes a real threat, a computer worm situation is the same as this. Predicting the propagation speed of network worms will help in analyzing the amount of threat a network worm is for a given network. In this paper, we provide:

- A brief presentation on network worms and the threat that it can pose to networks (Section 2)
- A presentation of the models that will be used to simulate the spreading process of network worms (Section 3)
- Presentation of implementations that were made to run simulations (Section 4)
- Results of simulation and thorough analysis (Section 5).

2 LITERATURE

A computer worm is a standalone malware computer program that replicates itself in order to spread to other computers. It often uses a computer network to spread itself, relying on security failures on the target computer to access it. It will use this machine as a host to scan and infect other computers. When these new worm-invaded computers are controlled, the worm will continue to scan and infect other computers using these computers as hosts, and this behavior will continue. Computer worms use recursive methods to copy themselves without host programs and distribute themselves based on the law of exponential growth, thus controlling and infecting more and more computers in a short time. The main threat of this malware is coming from its contagiousness. To illustrate this, here are some figures :

- the worm SQL Slammer was the fastest computer worm in history, the infected population doubled in size every 8.5 seconds [4].
- more than 359,000 computers connected to the Internet were infected with the Code-Red (CRv2) worm in less than 14 hours [5].

3 MODELS

Since spreading a network worm in the Internet is forbidden, networks models and spreading process of a network worm model are a crucial step in our work. Here we describe the models that will be used when implementing the experiment:

- Network model : Networks are composed of many devices such as routers, switches, hubs, laptops, mobile phones and so on. Each of these devices can be infected by a network worm and then be a part of the spreading process of a network worm. So, we use existing graph models to simulate networks, we associate the nodes of a graph to devices that composed real networks and we associate links between the nodes of a graph to the connections between real devices such as Wifi connection, Ethernet connection, Bluetooth connection and so on. First, we used Erdos-Renyi model [6], a model producing a graph with a given number of nodes and links, however with simulation, the results that were produced weren't satisfying since this model is not a great representation of real networks because

edges has a fixed probability of being present or absent. Second, we used Barabási–Albert model, this algorithm generates random scale-free networks using a preferential attachment mechanism meaning that the network follow a power law distribution [6]. This is the model of network that we finally choose since the Internet follow a power law distribution [3].

- Spreading process model SI [1, 2] : A node in the graph can have two status Susceptible (S) or Infected (I). An infected node can not go back to the Susceptible status. At each step (the time is discretized), all infected nodes try to infect all their Susceptible neighbor in the graph with a probability p representing the probability of the node (device) having vulnerability that enable the worm to infect it representing in reality that the link between the nodes is up or down.

4 IMPLEMENTATIONS

First step, we generate a random graph with the model of Erdos-Renyi. We implement the model with the code in Figure 2to produce a random graph with a given number of nodes and links, each edges having a fixed probability to exist.

```
def erdos(node_count, link_count):
    my_graph = {}
    for node in range(node_count):
        my_graph[node] = []
    current_link_count = 0
    while current_link_count <= link_count:
        node1, node2 = randint(0, node_count -1),
        ↪ randint(0, node_count -1)
        link_exists = node1 in my_graph and node2
        ↪ in my_graph
        link_exists = link_exists and node1 in
        ↪ my_graph[node2]
        link_exists = link_exists and node2 in
        ↪ my_graph[node1]
        if not link_exists:
            add_link((node1, node2), my_graph)
            current_link_count += 1
    return my_graph
```

Figure 1: Program implementing the generation of Erdos-Renyi graph

We create the visualization of the graph in Figure 2 however through the comparison with the topology of real networks, we drop the simulation of the spreading process on this graph.

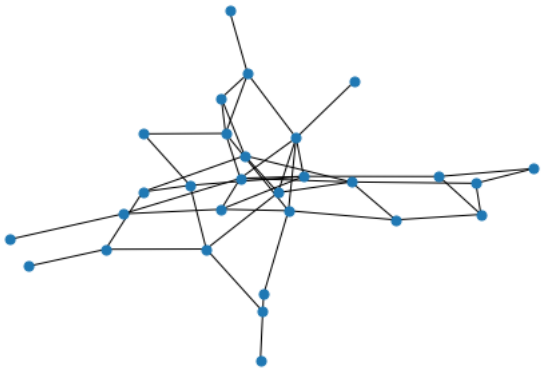

Figure 2: Erdos-Renyi graph

Second step, we generate a graph using the Barabasi-Abert model. Briefly, this program attach successively to an existing graph additional nodes with a specific degree α. This representation is implemented with the following program :

```
def barabasi(graph_size, my_graph, alpha):
    base_graph = copy.deepcopy(my_graph)
    origin_size = len(base_graph)
    for new_node in range(origin_size + 1,
    ↪ graph_size + 1):
        node_odd = []
        my_sum_degree = sum_degree(base_graph)
        cumul = 0
        for node in base_graph: # Compute the
        ↪ odds for each node to be linked to
        ↪ the new node
            cumul += len(base_graph[node]) /
            ↪ my_sum_degree
            node_odd.append((node, cumul))
        base_graph[new_node] = [] # New node
        ↪ added to graph
        degree_new_node = 0
        while degree_new_node < alpha:
            tmp = uniform(0,1)
            neighbour_node = None
            for (node, odd) in node_odd:
                if tmp < odd:
                    neighbour_node = node
                    break
            link_exist = neighbour_node in
            ↪ base_graph[new_node] and
            ↪ new_node in
            ↪ base_graph[neighbour_node]
            if not link_exist:
                add_link((new_node,
                ↪ neighbour_node), base_graph)
```

```
                degree_new_node += 1
    return base_graph
```

The topology generated in Figure 3 is similar to the Internet since there are many nodes at its center with high degre similar to routers with many connections, and nodes at the border connected to nodes with high degree similar to devices connected to routers.

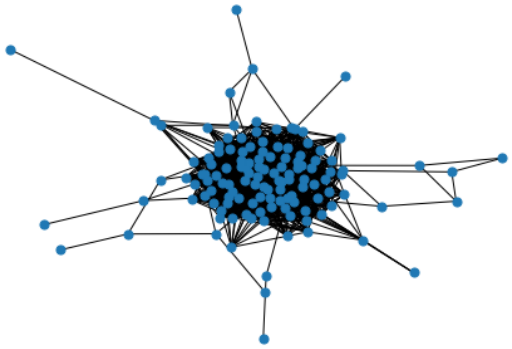

Figure 3: Barabasi-Albert graph

Final step, implementing the spreading process described in Section 3 on the graph generated in Figure 3, this is done by the code in the Figure 4 :

```
def si_model(lcc, my_graph, p):
    max_infected = len(lcc)
    infecteds = {lcc.pop()}
    step = 0
    my_evolution = {step : len(infecteds)}
    while len(infecteds) < max_infected:
        step += 1
        new_infected = set()
        for infected in infecteds:
            for neighbour in my_graph[infected]:
                tmp = uniform(0,1)
                if tmp < p:
                    new_infected.add(neighbour)
        infecteds = infecteds.union(new_infected)
        my_evolution[step] = len(infecteds)
    return my_evolution
```

Figure 4: Program implementing the spreading process

5 RESULTS AND ANALYSIS

The objective was to compute the variation of the propagation speed according to network structure. To compute it, we run several simulations with a Barabasi-Albert graph with 1000 nodes and α in a range from 1 to 10 and we plot the evolution of the time needed for all nodes to be infected according to the value of α

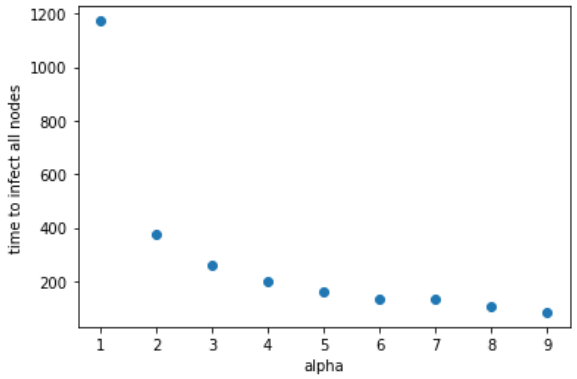

Figure 5: Time needed to infect all nodes according to α

The results in Figure 5 show that the time needed to infect all nodes reduce exponentially when the number of connection between nodes goes from 1 to 2, more specifically the times goes from 1190 to 390, only 32.7% of the original time. The speed reduction decrease. For instance when the value of α goes from 2 to 3, the time goes from 390 to 270, 70% of the original time.

6 CONCLUSION

In this study, we presented two models to simulate the spreading process of a network worm. We made an implementation with Python a programming language. We ran experiments and determined the speed variation of network worm according to the number of connection between devices. This work only considered a simple model of network, more realistic network modelling can be done in future work.

REFERENCES

[1] V. Blavatska and Yu Holovatch. 2021. Spreading processes in "post-epidemic" environments. II. Safety patterns on scale-free networks. *arXiv:2112.01431 [cond-mat]* (Dec. 2021). http://arxiv.org/abs/2112.01431 arXiv: 2112.01431.

[2] J Demongeot, Q Griette, and P Magal. 2020. SI epidemic model applied to COVID-19 data in mainland China. *R. Soc. Open Sci.* 7, 12 (Dec. 2020), 201878.

[3] Michalis Faloutsos. [n. d.]. Georgos Siganos siganos@cs.ucr.edu. ([n. d.]), 32.

[4] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. 2003-01. *The Spread of the Sapphire/Slammer Worm.* Technical Report. CAIDA, ICSI, Silicon Defense, UC Berkeley EECS and UC San Diego CSE.

[5] D. Moore, C. Shannon, and J. Brown. 2002-11. Code-Red: a case study on the spread and victims of an Internet worm. In *Internet Measurement Workshop (IMW)*. 273–284.

[6] Mark Newman. 2010. *Networks: An Introduction.* Oxford University Press, Inc., USA.