

Evading Detection During Network Reconnaissance

Ilias Belalis¹^a, Georgios Spathoulas²^b and Ioannis Anagnostopoulos¹^c

¹*Department of Computer Science and Biomedical Informatics, University of Thessaly,
2-4 Papasiopoulou st., Lamia, 35131, Greece*

²*Department of Information Security and Communication Technology,
Norwegian University of Science and Technology (NTNU), Mail Box 191, Gjøvik, NO-2815, Norway*

Keywords: Reconnaissance, Port Scanning, Detection, Evasion, Genetic Algorithm, Covert.

Abstract: Network security attacks have seen a significant increase in recent years. A remote attacker needs to understand the topology of the victim network and extract as much information as possible for the hosts of the network. The first step of a network attack is called reconnaissance and aims at gathering such information. In this paper, we analyze the detection of such activity through the use of machine learning classifiers. We identify which are the characteristics of reconnaissance activity that render it detectable and employ a heuristic approach to decide optimal values for such fields that can produce undetectable port scanning traffic. Based on those findings, a covert port scanning tool has been developed and made publicly available. The tool executes the reconnaissance step of an attack in a way that it can evade being detected.

1 INTRODUCTION

Network security (Marin, 2005) is a set of technologies that protects the integrity, confidentiality, availability of data, and usability of network infrastructure by preventing entry or proliferation within a network of a wide variety of potential threats. Network security involves creating a secure infrastructure for devices, applications, and users in order to work in a secure manner. Specifically, Network Security involves antivirus software, endpoint detection, and response (Kaur and Tiwari, 2021), application security, access control, network analytics, firewalls, intrusion detection systems (Kanika, 2013), VPN encryption, and more.


Port Scanning (De Vivo et al., 1999) is an integral part of active reconnaissance attacks. It is the primary source for information gathering about the infrastructure of a target network for the attackers. Port scanning identifies port availability by sending connection requests to a target computer and recording corresponding responses. Determining which ports are in use enables attackers to understand which applications and services are active on the target machine. From there, an intruder can check for vulner-


abilities and initiate his attack plan. Machine Learning algorithms are increasingly used in solving cybersecurity problems and mainly in the detection of attacker's activity in a system (Xin et al., 2018). Such algorithms have also been proposed to be used for the detection of standard port scanning activity and have shown high accuracy rates.


In the present paper, the detection of port scanning activity is analyzed and studied to understand the optimal approach for that. Additionally, the actual metrics that enable detection are identified and an approach to implement an evading scanning technique that can achieve undetected scanning of a network is presented. The main contributions of the paper are the following:

- Identification of which are the most accurate classification algorithms with respect to detection of port scanning attempts.
- Identification of which are the most statistically significant parameters of network traffic that can lead to the detection of port scanning attempts.
- Design and implementation of a covert port scanning tool that can evade detection from aforementioned algorithms.

The rest of the paper is structured as follows. Section 2 presents related work in the field of port scanning detection using machine learning algorithms. In

^a <https://orcid.org/0000-0002-0188-1925>

^b <https://orcid.org/0000-0003-2947-486X>

^c <https://orcid.org/0000-0002-0832-0522>

Section 3, various machine learning classification algorithms are tested, to decide upon their applicability to the problem. In Section 4, the results of the ML algorithms are analyzed and the most important features are identified. In Section 5, a heuristic approach is used to optimize the values of such feature, to minimize the detection probability. In Section 6, a covert port scanning tool that is able to evade detection is presented. Finally, Section 7 presents our conclusions.

2 RELATED WORK

Detecting Port Scanning attacks using Machine Learning algorithms is one of the most popular trends in the Cyber-defense field. Bertoli et al. in (Bertoli et al., 2021) described the AB-TRAP framework which consists of (i) the generation of the attack dataset, (ii) the bonafide dataset, (iii) training of the machine learning models, (iv) implementation of the models and (v) the performance evaluation of the model. They test AB-TRAP in two environments: LAN and the Internet. In both cases, has been achieved low resource utilization and the Decision Tree classifier provided the best performance for the training and realization phases. Wankhede in (Wankhede, 2019) provided anomaly detection using machine learning techniques. Specifically, various types of anomalies such as DoS attacks, DNS poisoning, and Port Scanning are presented and can be detected by using supervised machine learning techniques like feedforward neural networks. Balram et al. in (Balram and Wiscy, 2008) described the detection of TCP SYN scanning using packet counts and neural networks. Through their work, they investigated the effectiveness of using counts of various aggregated TCP control packets in detecting TCP SYN scanning. Also, a neural network was trained in order to capture normal as well as port scanning data. TCP SYN, SYN/ACK and FIN packets show definite patterns in their behaviour for legitimate connections. As regards the detection of TCP SYN scanning, a deviation from the normal behaviour is observed. Jirapummin et al. in (Jirapummin and Kanthamanon, 2002) proposed a hybrid neural network approach for IDS. In more detail, they presented an intrusion detection system based on Self-Organizing Maps (SOM) and Resilient Propagation Neural Networks (RPROP) for visualizing and classifying intrusion and normal patterns. Andropov et al. in (Andropov et al., 2017) presented network anomaly detection using artificial Neural networks. Through their work, they presented a method of identifying

and classifying network anomalies using an artificial neural network for analyzing data gathered via Net-flow protocol. The results showed high percentages of successful identifications after a number of iterations. Algaolahi et al. in (Algaolahi et al., 2021) used supervised machine learning classifiers in order to detect port scanning attacks. Specifically, their work is focused on detecting port scanning attacks by using different machine learning algorithms and comparing them to find the best one. As regards the results, some algorithms such as Decision Tree and Random Forest reach nearly 100 percent of accuracy with a short time of training and testing.

3 DETECTION OF RECONNAISSANCE ACTIVITY

As mentioned in the previous Section, various approaches exist in the literature that attempts to automatically detect reconnaissance activity. The majority of existing approaches address port scanning detection, as this enables attack mitigation in its early stages. As regards, port scanning detection, one solution can be traditional network intrusion detection systems which are complex and require resources and maintenance. An alternative approach could be based on lightweight machine learning classifiers, that upon being properly trained can provide a fast detection mechanism for reconnaissance network activity. The paper focuses on machine learning-based detection of reconnaissance activity and the evasion of detection from such approaches.

3.1 The Classifiers

In the present Section, five commonly used machine learning classifiers were picked and used to detect port scanning activity. The machine learning classifiers that were tested were: (a) Decision Tree (DT)(Kotsiantis, 2013), (b) Random Forest (RT)(Pal, 2005), (c) Multi-layer Perceptron (MLP)(Ramchoun et al., 2016), (d) k-Nearest Neighbours (kNN)(Viswanath and Sarma, 2011) and (e) Naive Bayes (NB)(Rish et al., 2001).

3.2 The Data-Set

The classifiers were trained and tested upon a dataset built through the use of the AB-TRAP Project¹. The aforementioned project allows the integration of realistic background traffic with traffic produced by port

¹<https://github.com/c2dc/AB-TRAP>

Table 1: TCP port scanning tools and techniques.

Port scanning tools	TCP Port scanning techniques
Nmap	SYN, Connect, FIN, XMAS, NULL, ACK, Window
Unicornsca	SYN, Connect, FIN, XMAS, NULL, ACK
Hping3	SYN, Connect, FIN, XMAS, NULL, ACK
Zmap	SYN
Masscan	SYN

scanning tools. Specifically, the attack network traffic consists of Port scanning tools and techniques as described in Table 1. Regarding the normal traffic dataset, the MAWILab dataset (Fontugne et al., 2010) has been used. It is 15 minutes of daily traffic from a transpacific backbone between the USA and Japan. The data-set used consists of 416,249 packets in total, out of which 100,520 have been produced by reconnaissance activity, while the rest 315,729 correspond to normal traffic.

Features of the traffic that were identified as not generic enough for this study were removed. Examples of such features are link layer (layer 2) fields and features that are redundant or invariable such as source IP address, destination IP address, IP version, etc. Table 2 shows the fields that were fed to the classifiers.

3.3 Classification Results

Using the described data-set, the classifiers were assigned a supervised learning task in the form of a binary classification problem; classifying packets as "Attack" (reconnaissance activity) or "Normal" (background traffic). The data-set was split into training and test sets at a ratio of 70-30%. Table 3 presents the accuracy ratio per classifier.

The results show that most of the classifiers (all apart from Naive Bayes) achieve very high accuracy ratios approximately equal to 99%.

4 ANALYSIS OF RECONNAISSANCE TRAFFIC

In the present Section, further analysis of the results of the experiments described in the previous Section is attempted. The goal of this analysis is to understand the data upon which port scanning traffic is detectable by most algorithms with a very high accuracy ratio. To achieve this we need to find the fields (features) of the IP packets which were the most statistically signifi-

cant for classifiers to achieve the final result. For each of the algorithms used, we analyzed the features' importance and concluded on the most significant fields. This analysis has been conducted upon the mean decrease impurity approach (Li et al., 1984).

Table 4 depicts the importance of all features that have non negligible effect for at least one of the used classifiers. However, for the MLP classifier it was not possible to calculate the most important fields and it is not included in the Table 4.

The results shown in Table 4 enable us to conclude on a minimal set of features which allow classifiers to successfully detect the port scanning activity. Those fields are:

- **ip.ttl:** Tools tend to pick a specific ttl value for the packets they send and this enables classifiers to identify such packets.
- **tcp.srport:** Scanning tools tend to pick default ports as source ports for the packets they send and thus those are easily detected.
- **tcp.window_size:** The packets sent have specific sizes as well, which makes such detectable.
- **tcp.seq:** The TCP sequence number gets default values which make the packets detectable.

5 EVADING DETECTION

In this section, an approach to implement a port scanning approach that can evade detection by a trained classifier is discussed. In other words, the covert port scanning tools proposed attempt to conduct reconnaissance steps while the packets it produces are not classified as "Attack" by the classifier.

The main concept for implementing such a tool is to build upon the important fields identified in the previous Section. Choosing appropriate values for the four fields discussed could enable the tool to evade detection. In practice, the packets produced by the scanning tool shall resemble more to the packets of normal traffic (at least for the significant fields). So the main task is to optimize the values of those fields towards lowering the detection probability by all classifiers.

This optimization problem has been approached through the use of genetic algorithms. The problem to be solved is to select the optimal values for the packet fields, identified as the most significant ones, to minimize the probability for the reconnaissance activity to be detected. The large size of the search space (all possible values for the four fields) prohibits the exhaustive search approach. Hence, a heuristic optimization method has to be employed (Rothlauf, 2011), so we selected to use a genetic algorithm.

Table 2: Packet Fields/ ML Features.

Packet Field	Description
ip.id	The IP identifier (IP-ID) is a 16 (32) bits field in the IPv4 (v6) header.
ip.flags.df	When set, the df (Don't Fragment) indicates that the packet cannot be fragmented for transmission.
ip.ttl	The TTL field, Time To Live, of an IP packet represents the maximum number of IP routers that the packet can go through before being discarded.
ip.len	Specifies the length of the IP packet that includes the IP header and the user data.
ip.dsfield	The Differentiated Services field marking packets for Different Quality-Of-Service (QoS) levels. For example, data belonging to voice and video protocols have no acceptance for the delay.
tcp.srcport	Identifies the sending port.
tcp.seq	TCP is a byte-oriented sequencing protocol. A Sequence Number field is necessary to ensure that missing or misordered packets can be detected and fixed.
tcp.len	The TCP payload size is calculated by taking the "Total Length" from the IP header (ip.len) and then subtract the "IP header length" (ip.hdr.len) and the "TCP header length" (tcp.hdr.len).
tcp.hdr_len	A 4-bit field containing the length of the IP header in 32-bit increments.
tcp.flags.fin	Last packet from sender.
tcp.flags.syn	Synchronize sequence numbers. Only the first packet sent from each end should have this flag set. Some other flags and fields change meaning based on this flag, and some are only valid when it is set, and others when it is clear.
tcp.flags.reset	Reset the connection.
tcp.flags.push	Asks to push the buffered data to the receiving application.
tcp.flags.ack	Indicates that the Acknowledgment field is significant. All packets after the initial SYN packet sent by the client should have this flag set.
tcp.flags.urg	Indicates that the Urgent pointer field is significant.
tcp.flags.cwr	Congestion window reduced flag is set by the sending host to indicate that it received a TCP segment with the ECE flag set and had responded in congestion control mechanism.
tcp.window_size	The size of the receive window, which specifies the number of window size units that the sender of this segment is currently willing to receive.
tcp.urgent_pointer	If the URG flag is set, then this 16-bit field is an offset from the sequence number indicating the last urgent data byte.
tcp.options.mss_val	The Maximum Segment Size (MSS) is a TCP Option and sets the largest segment that the local host will accept.

Table 3: Accuracy ratio per classifier.

ML Classifier	Accuracy Ratio
Decision Tree (DT)	0.9903
Random Forest (RF)	0.9902
Multi-layer Perceptron (MLP)	0.9906
k-Nearest Neighbours (kNN)	0.9867
Naive Bayes (NB)	0.4288

The approach used was to employ the pre-trained classifiers and then construct typical scanning packets (for which the values of the four significant fields were selected heuristically). Each selection of values corresponds to a traffic packet, which can be fed to the trained classifiers, and upon the probability that the packet is identified as port scanning, we can conclude

Table 4: Features Importance.

Feature	DT	RF	kNN	NB
ip.ttl	0.954	0.367	0.000	0.000
ip.dsfield	0.025	0.000	0.000	0.000
tcp.options.mss_val	0.011	0.000	0.057	0.000
tcp.hdr_len	0.000	0.111	0.000	0.000
tcp.window_size	0.000	0.101	0.181	0.000
tcp.srcport	0.000	0.000	0.174	0.000
tcp.seq	0.000	0.000	0.000	0.089

the fitness of the selection of values. The workflow is depicted in Figure 1.

The design parameters of the genetic algorithm

are as follows:

- The search space comprises all possible combinations of values for the four fields. Because the candidate values for fields, such as port, size, or ttl are actually large ranges of integer values (most of which may be invalid or non-applicable) we constrained the search space according to the most used values. By analyzing the traffic data-set, we picked the most 20 common values for each field, thus creating a search space with a solutions population $p = 20^4$.
- Each solution is represented by an array of four elements denoted as f that represents the value assigned to each field.
- The fitness function is defined as:

$$fit(f) = \frac{\sum_{i=1}^N prob_i(f)}{N}$$

, where $prob_i(f)$ is the probability that a packet constructed with the values of f is classified as port scanning activity by classifier i and N is the total number of classifiers.

- The initial population size is 100.
- The mutation probability is 0.1.
- The next generation is determined by uniform crossover, with crossover probability equal to 0.5, an elite ratio of 0.01, and 0.3 of the population consisting of the fittest members of the previous generation (aka parents).
- The algorithm terminates when the maximum number of allowed iterations is used. This number is calculated as $itermax = 50 * p$.

In order to select the most appropriate values, a program has been developed in Python. The main functions of the program are:

- Data structure with values for the most important fields such as IP TTL, TCP Source Port, and TCP Window Size as captured in "Normal" network traffic.
- Using a Genetic Algorithm that feeds with the above values each one of the ML classifiers such as Decision Tree, Random Forest, Multi-layer Perceptron, Naive Bayes, and k-Nearest Neighbours.
- The Genetic Algorithm returns the values that will have the highest average score for the above classifiers.

The implementation of the above mechanism is shown in Figure 1. By forging a new packet with the values, which have been calculated by the Genetic

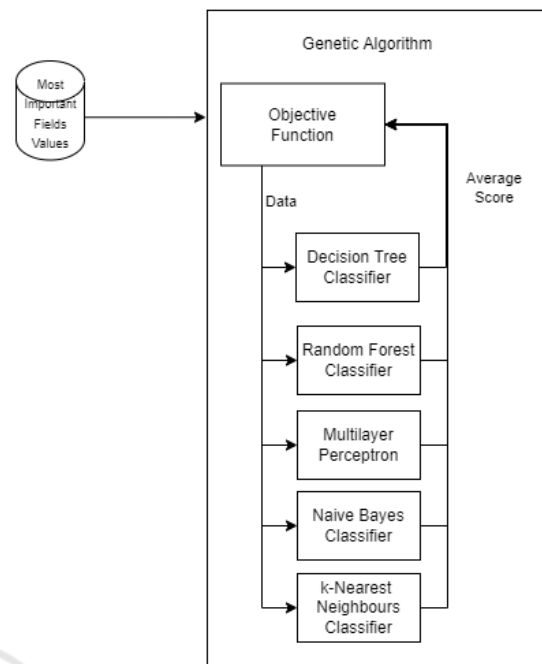


Figure 1: Genetic Algorithm.

Algorithm, it is possible to evade detection for Port Scanning.

The approach proposed is based on the idea of picking values for fields of the packets (that can be manipulated) which will make detecting port scanning activity harder. Of course, it is feasible to train classifiers upon the packets produced by our approach, but a dynamically configured port scanning tool can successfully evade detection, as retraining classifiers would not be an easy measure to take.

6 COVERT NETWORK SCANNING TOOL

In this section, we present a covert port scanning tool that was developed in Python through the use of the Scapy Framework. The tool performs a number of port scanning techniques and through the use of the values calculated in Section 5 it is able to evade detection by all classifiers discussed in the previous Sections. The tool and its source code are available at GitHub².

The tool implements TCP SYN Scan, TCP Connect Scan, TCP NULL Scan, TCP FIN Scan, and TCP XMAS Tree Scan and enables the user to retrieve information about a network, without being detected.

To validate this claim we went through a number

²<https://github.com/ilbe753/Simple-Port-Scanner>

of experiments. Specifically, we performed different port scanning attempts for different types of scanning (TCP SYN Scan, TCP Connect Scan, TCP NULL Scan, TCP FIN Scan, and TCP XMAS Tree Scan). For each of the above port scanning attempts, the network traffic was captured, through the use of *wireshark*³.

We trained a number of classifiers (DT, RF, MLP, kNN, NB) according to the analysis presented in Section 3. To make the experiment more realistic, we trained those classifiers with multiple combinations of real-world background traffic and traffic chunks produced by a number of port scanning tools, such as *nmap*⁴, *zmap*⁵, *masscan*⁶, and *hping*⁷.

The covert scanning tool was used to conduct a number of reconnaissance attempts with various techniques. The activity of the covert scanning tool was not detected in any of the scenarios. For all combinations of the (a) classification algorithm, (b) scanning tool that has been used for training the classifier and (c) scanning technique used by the covert tool, the activity of the latter remained undetected.

7 CONCLUSIONS

In this paper, the detection of reconnaissance activity through the use of ML classifiers has been studied. Literature was analyzed and both efficient algorithms and use-full network packet fields to the process were identified. The extracted information was used to train a number of classifiers in order to detect port scans with high accuracy. This has confirmed that it is feasible to detect port scanning activity with this approach.

Consequently, the most significant packet fields that enable high accuracy ratio metrics for most of the algorithms were identified a genetic algorithm approach was used to heuristically decide the optimal values for such fields that would enable port scanning activity while remaining undetected by classifiers. Based on those findings, a covert port scanning tool was developed and made publicly available for the network security research community. The tool was tested under various circumstances and it has always evaded detection.

As future work plans, we foresee that we can add dynamic updates of the evasion capabilities of the proposed tool, according to novel scanning

tools/algorithms. Re-defining the significant fields and the proper values for those may be done centrally and then updating the configuration parameters of all instances of the covert scanning tool through an over-the-air update.

REFERENCES

- Algaolahi, A. Q., Hasan, A. A., Sallam, A., Sharaf, A. M., Abdu, A. A., and Alqadi, A. A. (2021). Port-scanning attack detection using supervised machine learning classifiers. In *2021 1st International Conference on Emerging Smart Technologies and Applications (eSmarTA)*, pages 1–5. IEEE.
- Andropov, S., Guirik, A., Budko, M., and Budko, M. (2017). Network anomaly detection using artificial neural networks. In *2017 20th Conference of Open Innovations Association (FRUCT)*, pages 26–31. IEEE.
- Balram, S. and Wiscy, M. (2008). Detection of tcp syn scanning using packet counts and neural network. In *2008 IEEE International Conference on Signal Image Technology and Internet Based Systems*, pages 646–649. IEEE.
- Bertoli, G. D. C., Júnior, L. A. P., Saotome, O., Dos Santos, A. L., Verri, F. A. N., Marcondes, C. A. C., Barbieri, S., Rodrigues, M. S., and De Oliveira, J. M. P. (2021). An end-to-end framework for machine learning-based network intrusion detection system. *IEEE Access*, 9:106790–106805.
- De Vivo, M., Carrasco, E., Isern, G., and De Vivo, G. O. (1999). A review of port scanning techniques. *ACM SIGCOMM Computer Communication Review*, 29(2):41–48.
- Fontugne, R., Borgnat, P., Abry, P., and Fukuda, K. (2010). Mawilab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *Proceedings of the 6th International Conference, Co-NEXT '10*, New York, NY, USA. Association for Computing Machinery.
- Jirapummin, C. and Kanthamanon, P. (2002). Hybrid neural networks for intrusion detection system. In *Proceedings of the IEEK Conference*, pages 928–931. The Institute of Electronics and Information Engineers.
- Kanika, U. (2013). Security of network using ids and firewall. *International Journal of Scientific and Research Publications*, 3(6):1–4.
- Kaur, H. and Tiwari, R. (2021). Endpoint detection and response using machine learning. In *Journal of Physics: Conference Series*, volume 2062, page 012013. IOP Publishing.
- Kotsiantis, S. B. (2013). Decision trees: a recent overview. *Artificial Intelligence Review*, 39(4):261–283.
- Li, B., Friedman, J., Olshen, R., and Stone, C. (1984). Classification and regression trees (cart). *Biometrics*, 40(3):358–361.
- Marin, G. A. (2005). Network security basics. *IEEE Security & privacy*, 3(6):68–72.

³<https://www.wireshark.org/>

⁴<https://nmap.org/>

⁵<https://github.com/zmap/zmap>

⁶<https://github.com/robertdavidgraham/masscan>

⁷<https://www.kali.org/tools/hping3/>

- Pal, M. (2005). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222.
- Ramchoun, H., Ghanou, Y., Ettaouil, M., and Janati Idrissi, M. A. (2016). Multilayer perceptron: Architecture optimization and training.
- Rish, I. et al. (2001). An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46.
- Rothlauf, F. (2011). *Design of modern heuristics: principles and application*, volume 8. Springer.
- Viswanath, P. and Sarma, T. H. (2011). An improvement to k-nearest neighbor classifier. In *2011 IEEE Recent Advances in Intelligent Computational Systems*, pages 227–231. IEEE.
- Wankhede, S. B. (2019). Anomaly detection using machine learning techniques. In *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, pages 1–3. IEEE.
- Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., Gao, M., Hou, H., and Wang, C. (2018). Machine learning and deep learning methods for cybersecurity. *IEEE Access*, 6:35365–35381.

