# Automating synthetic generation and capture of network scanning packet captures

Ørjan Alexander Jacobsen
*Cyber Security*
*Noroff University*
Bodø, Norway
oaj@oaj.guru

***Abstract*—Today the lack of early warning scanning detection is a problem needed to be mitigated. This article will explain the evolution and historical background of network scanning methods from the early internet age until today. People today use network scanning for multiple purposes; conducting audits towards their own manageable infrastructure, researching purposes, conducting cyber attacks, or autonomous malicious activity. The importance of the knowledge of how a network scanner works and various types of scans will be elaborated on within this research.**

**Related to the pending security action being conducted, a base knowledge of which types of tools to conduct various tasks such as audit is crucial to mitigate risks of network scanning. Classifying network activity is an important step in mitigating future risks of cyberattacks. In order to detect such activity, synthetic data sets are needed to compare synthetic generated network scanning packet captures to real-world scanning data. This paper will focus on the setup of a virtual lab environment, where synthetic scanning data would be generated automatically through a tasking system and prepared for comparative analysis.**

***Index Terms*—Classification, detection, nmap, network scanning**

## I. Introduction

Scanning attacks are often applied to collect information about a network, mainly in the first step of the cyber kill chain known as the reconnaissance phase. The Lockheed Martin Cyber Kill Chain [1] is a framework used for identification and prevention of cyber intrusions activity, which is a part of the Intelligence Driven Defense model. The Cyber Kill Chain framework proves seven phases of a kill chain to accomplish a successful attack. Threats should ideally be detected within the first steps of the kill chain; reconnaissance, weaponization, and delivery [2]. The report distinguishes between two-phase detections; early and late phase detection. In the late phase, the detection is within the step of command and control, which is the 6th step in the kill chain. This means the threat actor already has delivered and installed malicious components on the target side. Meanwhile, in the early phase, the detection is noticed within the delivery phase, where threat actors attempt to deliver the malicious component towards a target.

Scanning is also a central part of penetration testing, where vulnerabilities are identified. During the phase of scanning and information gathering, activities such as port and vulnerability scanning are conducted as a normal thing for gathering intelligence.

Understanding patterns of attacks and threats are crucial to allow early detection of attacks [3]. Primarily detecting known scanners is an important step of both ongoing detecting scans and preventing cyber attacks. Analyzing amounts of data allows for gaining an understanding of patterns used in attacks [3]. The main objective of this research is to automate network scans in order to generate synthetic data sets, which in the future would optimize the method of collecting quality data for analysis purposes. The research outcome is expected to generate data sets containing synthetic data, which would be used as a step in the preparation process of signatures as a final research output.

This paper elaborates on the automatic generation of synthetic network scanning packet captures using various types of scan modes and templates using the Nmap scanner. By generating synthetic data sets, where the control of the tasks conducted against each worker host running packet capturing, the analysis of these data sets could be used in a comparative analysis to classify and detect various types of network scans. Following this method elaborated in the paper would allow synthetic data generation for use in a comparative analysis.

This paper is structured into sections elaborating related work in section II, the methodology of generating and collection in section III-B, and processing of the synthetic packet captures in a virtual lab environment. The needed software, operating systems, and packages are further elaborated in section III-A.

## II. Related work

Network scanning is a reconnaissance activity conducted within the field of network intrusion [4]. During an intelligence-gathering step, network scanning is one of the first steps before an attack is about to happen [5]. Conducting a network scan would reveal information about hosts on a network, which includes running services and applications, open ports, and operating system details. The main techniques applied during network scanning are version and service detection, scanning ports, operating system detection, and network mapping [6]. During a network scan, devices and services vulnerable to attacks could be unveiled [5].

By conducting port scanning, information about running services and open ports could be retrieved. This could further be used to determine misconfiguration on a system or version of the software, which could be mapped against known exploitable vulnerabilities. Multiple types of port scanning methods exist, such as TCP SYN and XMAS. Another interesting scan is an ICMP echo scan [6].

Port scanning is a reconnaissance technique used to discover services running on open ports which could be exploited to gain access to a system. Port scanning is primarily divided into vertically and horizontally scans [7].

In vertical attacks, multiple ports are scanned on one host. This is a common process when a threat actor is interested in one specific host to attack. Vertical scanning sends probes on various ports against one specific host on the network [4]. A vertical scan could be used to map out known vulnerabilities using a scanner to detect various operating system and service information, including versioning information.

Nmap is free software released in September 1997, mainly created for users to scan networks under security audits by system administrators and other IT professionals. It has a wide range of capabilities, such as port scanning a single host or a subnet, detecting operating system and service versioning information, and echo ping scans. The software has various usage areas, such as asset management, system, and network inventory, compliance testing, and security auditing. During audits, it could be useful to conduct OS versioning to detect the need for patching and updates. Since Nmap could usually be used for this purpose, it could also be used during harmful intrusions to map the infrastructure of a network. Furthermore, this could be used to exploit vulnerabilities within the network. Regarding types of scans that could be conducted, Nmap could use various timing templates to avoid detection, such as paranoid and sneaky mode. On the other side, Nmap can speed up a scan by applying aggressive or insane scan since these templates only are based on timings [6].

In cyberspace today, threats have become a major concern for businesses. There exist other security mechanisms than firewall and anti-virus that need to be implemented to increase security within a company's network [8]. Network scanning is a common reconnaissance activity [4], and the classification of a network scanner could be described. Furthermore, illustrating the complexity of the activity shows that over 96% of all anomalous events can be detected and labeled. These results would make the unlabelled source rate very low [9].

For the mission of detecting abnormal activity within a network, a taxonomy of anomalies could be created to detect such behavior. There exist multiple techniques for detection, such as creating signatures for pattern recognition statistical methods, among other things. These methods provide information that further could be used for anomaly detection. Traffic metrics could be used for detecting anomalies, such as scanning activity and DDoS. Labeling each occurring event could be applied to categorize and differentiate between abnormal traffic and normal traffic. Categorization of traffic could be separated into subcategories such as unknown, anomalies, or normal [10].

The observation of cyber attacks and anomalies has among security researchers enabled Liu and Fukuda to develop cyberspace monitoring approaches to detect anomalies. The study also shows that darknet provides effective passive monitoring approaches, which often refer to globally routable and unused IP addresses that are used to monitor unexpected incoming network traffic. This is an effective approach for viewing network security activities remotely. The main goal of the study was to create and propose a simple taxonomy of darknet traffic [11].

Mazel, Fontugne, and Fukuda documented anomalous events classified using data collected through six years [10]. The research points out that previously unknown events are now accountable for a substantial number of UDP network scans, scan responses, and port scans. With their proposed taxonomy, the number of unknown events has decreased from 20% to 10% of all events compared to the heuristic approach [10]. Such scans can be handled by the Nmap scanner, popularly used for administration, security auditing, and network discovery [6]. Subverting firewalls and IDS, optimization of Nmap performance, and automating common network tasks could also be done with the Nmap.

Scans are one type of event among others, which could be represented through characteristics and patterns. It might be several or single scanned hosts. Scans are conducted to gain knowledge about a target. This can be opened ports, which operating system is running, or versions of software that are using the open ports. In a network scan, hosts alive could be detected among services and ports open. During a host scan, the same results could be retrieved through checks towards the allowance of packets to a specific port. Network scans can be used to gather intelligence of how the network is structured, how many clients are connected, and which services at which version are located on the network. After gathering intelligence about clients in a network, the intelligence itself can tell something about which exploitation could be used against the various services at which specific version [10] during the delivery step in the cyber kill chain.

Various traffic characteristics during scanning would be caught, such as the following. During TCP scanning, SYN is used to initiate connection while ACK is mapping ports by filtering port answers through ICMP [10]. A signature is created with rules for different attributes of the traffic. Abnormal characteristics such as scanning attempts can be detected and generate a signature out of the behavior of the traffic recently conducted [10].

While identifying scanning techniques, a quantitative amount of data is required to conduct identification and evaluation of scans conducted. During the research of [4] the port scanning tool, Nmap was used to construct the scanning, and tcpdump was used to capture packets on the network interface of a FreeBSD host.

During research by Jammes and Papadaki [12], Snort was able to detect five out of six of the different timing templates used for Nmap, which is default delivered by Nmap. The last template, the paranoid template, was Snort not able to detect, according to the study. These templates are named insane, aggressive, normal, polite, sneaky, and paranoid. The difference between these is the minutes waited between sending a probe to the target host. During the paranoid mode, Nmap waits 5 minutes, while at the sneaky mode, it waits for 15 seconds [6], [12]. During a scan execution, the parameter $-T$ could be used to set the template. The higher the number is, the faster the scan are [6].

During a Nmap scan, other parameters could also be applied, such as the Time-to-Live parameter. Another method for avoiding detection is to randomize the order hosts are scanned. Nmap has capabilities for sending UDP and TCP packets with invalid checksums, which often firewalls and other security controls drop due to the fact the checksum isn't checked. If a reply is given, these are often from firewalls or other security controls.

## III. Generating Synthetic Data

To conduct this research, a virtual lab environment was needed to segment captured scanning traffic from real internal network traffic to ensure the validity of the synthetic generated packet capture data, meanwhile having the capability of managing each worker host. The importance of generating synthetic data is to have a clean data set, with minimal noise, for comparison against real-world data containing a substantial amount of noise traffic.

### A. *Lab environment*

VMware workstation 16.1 was used to set up 20 virtual hosts (referred to as workers) using Ubuntu 20.04 (Linux kernel 5.4.0-81-generic #91) with the only target of capturing packet captures from the scanner host. One virtual host (referred to as scanner host) were installed with Kali Linux 2021.4 (Linux kernel 5.14.0-kali4-amd64 #1) for conducting various type of scans, issuing tasks to the workers, and monitoring ongoing scans.

Two closed internal virtual networks are set up for the purpose of data generation and management. This is for the production of raw packet captures in accordance with generating comparative data sets, which could be used for comparison between the generated synthetic data and data in the wild. Through such a comparison, these synthetic data sets could be used to classify various types of scanners and scanning types. The chosen IPv4 unicast address block for virtual network 1 (referred to as "scanning network") is $192.168.2.0/24$. This address block is reserved for testing and is enlisted as TEST-NET-1 in RFC 5737 [13], which describes the usage of the network during this research. Each worker host has its increment asset name where the hostname is in accordance with the last two digits in the last IP octet (e.g., $bsc01$ resolving to $192.168.2.101$). The scanning host has a higher last IP octet to ensure no confusion and IP collision regarding the static allocation for worker hosts. Within this research, the scanning host has the address $192.168.2.230$. This address allocation logic only shows flaws when the number of workers is above 99. As this is a proof of concept and the low number of workers, this is not a problem.

Secondly, another virtual network is set up for the usage of managing worker hosts in the network. The main reason is to reduce noise traffic on the scanning network and to segment management of each worker, such as managing tasks, monitoring ongoing tasks, and retrieving raw packet capture data. Chosen IPv4 unicast address block for the virtual network 2 (referred to as "management network") is $194.100.10.0/24$. Within RFC 1466 [14] a class C network in the address space $194.0.0.0$ - $195.255.255.255$ is allocated for management use in Europe. The same logic for address allocation used in the scanning network is also applied to the management network, where the last IP octet represents each worker host. Both these networks are visualized in 1.

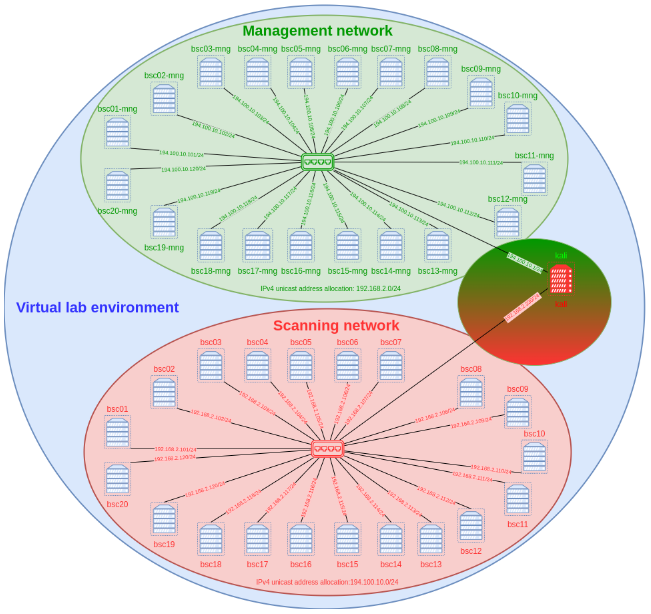


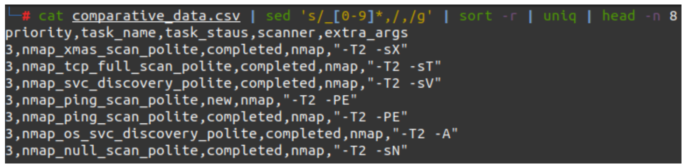
Fig. 1. Capture of network diagram for the virtual environment.

Another network characteristic is DHCP, which in both networks are disabled to ensure the logical structural setup of the network, such as static address allocation to each virtual host in accordance with a hostname. Basing the network setup on the RFC standards reduces the possibility for collision and confusion regarding usage of the networks [13], [14].

Through an initiation worker preparation script, there are implemented commands to reduce traffic noise affecting the scan results. This script disables certain services known for querying servers on the internet for the purpose of updating time through NTP, automatic operating system updates, and operating system package updates. This script must be run during the setup of the environment to ensure noise reduction. Another very important step during this initial configuration is change moving the tcpdump command to pecial ($chmod + s$), meaning a regular user is able to run tcpdump as a regular user. The command must, in this case, be executed by root. During the initial configuration of each worker, the SSH public key for the scanner host is added into the worker's authorized keys in $/.ssh/authorized\_keys$. It allows the scanner host to remotely access each worker with the use of its SSH private key to managing tasks for the worker through SSH. Each worker has its own $bscadm$ regular local user for this purpose. SSH listens only on the management network interface on each worker.

## B. Data generation and collection

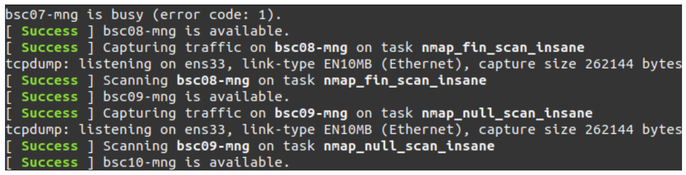
Management of each worker is conducted from the scanner host, as this is the host initiating the scanning tasks. The task issuing is conducted through a Bash script, running as a cron job on the scanner, iterating through given workers, and determining the first available worker. The task list is enlisted in a comma-separated file for enhanced readability for the user populating the task list as seen in figure 2.



```
└─# cat comparative_data.csv | sed 's/_[0-9]*,/,/g' | sort -r | uniq | head -n 8
priority,task_name,task_staus,scanner,extra_args
3,nmap_xmas_scan_polite,completed,nmap,"-T2 -sX"
3,nmap_tcp_full_scan_polite,completed,nmap,"-T2 -sT"
3,nmap_svc_discovery_polite,completed,nmap,"-T2 -sV"
3,nmap_ping_scan_polite,new,nmap,"-T2 -PE"
3,nmap_ping_scan_polite,completed,nmap,"-T2 -PE"
3,nmap_os_svc_discovery_polite,completed,nmap,"-T2 -A"
3,nmap_null_scan_polite,completed,nmap,"-T2 -sN"
```

Fig. 2. Capture of task list.

Given task files are iterated until a task with the status new is found. From this point, the scanner host deploys a task to the worker through the management network, commanding it to dump traffic on the incoming interface pointed to the scanning network. When this task is successfully deployed, the scanner hosts start the scan on the scanning network towards the worker with the given scanning characteristics. As a default, using the Nmap scanner, the result of the scan from the scanner host is outputted to an XML result file represented with the given task name. The status for the representative task in the task file is changed from $new$ to $ongoing$. This operation is iterated through all workers until all workers are busy when the scanning host cannot deploy more tasks until one or more of the workers are available for work again. A capture of this is shown in 3.

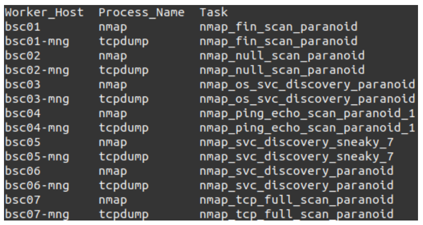


```
bsc07-mng is busy (error code: 1).
[ Success ] bsc08-mng is available.
[ Success ] Capturing traffic on bsc08-mng on task nmap_fin_scan_insane
tcpdump: listening on ens33, link-type EN10MB (Ethernet), capture size 262144 bytes
[ Success ] Scanning bsc08-mng on task nmap_fin_scan_insane
[ Success ] bsc09-mng is available.
[ Success ] Capturing traffic on bsc09-mng on task nmap_null_scan_insane
tcpdump: listening on ens33, link-type EN10MB (Ethernet), capture size 262144 bytes
[ Success ] Scanning bsc09-mng on task nmap_null_scan_insane
[ Success ] bsc10-mng is available.
```

Fig. 3. Capture of task allocation.

Aside from this, a task processing script is running to maintain the task list with the correct given status of tasks. This meaning is only comparing running scan processes, identified by task name on the scanning host, against running tcpdump tasks on the given worker. When this differs, and only the tcpdump process together with the task name are found in the process list, it terminates the tcpdump on the worker and updates the task status in the task list.

A monitor script is also created to monitor the ongoing tasks. This monitor updates every 2 seconds, capturing scanning specific tasks and tcpdump tasks returning data for worker corresponding with task name, shown in figure 4.



```
Worker_Host  Process_Name  Task
bsc01        nmap          nmap_fin_scan_paranoid
bsc01-mng    tcpdump       nmap_fin_scan_paranoid
bsc02        nmap          nmap_null_scan_paranoid
bsc02-mng    tcpdump       nmap_null_scan_paranoid
bsc03        nmap          nmap_os_svc_discovery_paranoid
bsc03-mng    tcpdump       nmap_os_svc_discovery_paranoid
bsc04        nmap          nmap_ping_echo_scan_paranoid_1
bsc04-mng    tcpdump       nmap_ping_echo_scan_paranoid_1
bsc05        nmap          nmap_svc_discovery_sneaky_7
bsc05-mng    tcpdump       nmap_svc_discovery_sneaky_7
bsc06        nmap          nmap_svc_discovery_paranoid
bsc06-mng    tcpdump       nmap_svc_discovery_paranoid
bsc07        nmap          nmap_tcp_full_scan_paranoid
bsc07-mng    tcpdump       nmap_tcp_full_scan_paranoid
```

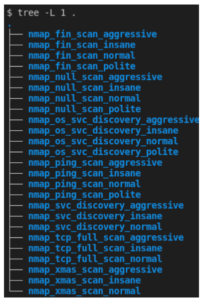
Fig. 4. Capture of scan monitoring tool

Other useful tools are developed, such as the task populator. The populator iterates through a given task file containing tasks that wanted to be run a number of times in order to generate a comparable synthetic data set in the end. An incrementing number is added to the task name (e.g., $nmap\_xmas\_scan\_1$) to uniquely identify a task when iterated through by the scan deployment stage.

A simplistic collection of captured packet captures run through a retrieve script on the scanning host, which iterates through each worker host matching a specific filter (e.g., task name) and remotely synchronizes packet captures to the local output directory on the scanning host.

## IV. Data processing

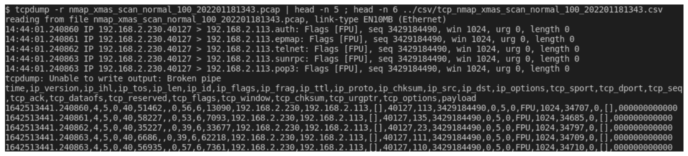
One of the components developed in this research is the packet capture parser. Inputs to this parser are the result directory containing packet captures captured by each worker during each separate scan task, and also an IP ignore list to filter out unwanted IP addresses. The retrieval component synchronizes packet captures to an output directory given by an input parameter. To keep a structure to separate each task from each other for use in the analysis, the directory tree structure for the retrieved packet capture is shown in figure 5.



```
$ tree -L 1 .
.
├── nmap_fin_scan_aggressive
├── nmap_fin_scan_insane
├── nmap_fin_scan_normal
├── nmap_fin_scan_polite
├── nmap_null_scan_aggressive
├── nmap_null_scan_insane
├── nmap_null_scan_normal
├── nmap_null_scan_polite
├── nmap_os_svc_discovery_aggressive
├── nmap_os_svc_discovery_insane
├── nmap_os_svc_discovery_normal
├── nmap_os_svc_discovery_polite
├── nmap_ping_scan_aggressive
├── nmap_ping_scan_insane
├── nmap_ping_scan_normal
├── nmap_ping_scan_polite
├── nmap_svc_discovery_aggressive
├── nmap_svc_discovery_insane
├── nmap_svc_discovery_normal
├── nmap_tcp_full_scan_aggressive
├── nmap_tcp_full_scan_insane
├── nmap_tcp_full_scan_normal
├── nmap_xmas_scan_aggressive
├── nmap_xmas_scan_insane
└── nmap_xmas_scan_normal
```

Fig. 5. Capture of tree directory structure for retrieved packet captures.

The parser then iterates through the given result directory identifying each respective task by name and the respective files related to the task. Scapy is used to read each representative packet capture into an object. Furthermore, header fields are defined for IP, TCP, UDP, and ICMP header, which are directly written to a file, each resulting in 3 files (TCP, UDP, and ICMP), with the pattern enlisted as *protocol_taskname.csv*. Each packet is iterated through, where only IP packets are processed. The parser creates a list of each packet during the iteration and detects the protocol before writing the packet details into its representative CSV file. Another Python library used during iteration is the binascii library, which decodes the payload to hex. A comparison of the raw packet capture and the parsed CSV file is shown in figure 6.



```
$ tcpdump -r nmap_xmas_scan_normal_100_202201181343.pcap | head -n 5 ; head -n 6 ../csv/tcp_nmap_xmas_scan_normal_100_202201181343.csv
reading from file nmap_xmas_scan_normal_100_202201181343.pcap, link-type EN10MB (Ethernet)
14:44:01.240860 IP 192.168.2.230.40127 > 192.168.2.113.auth: Flags [FPU], seq 3429184490, win 1024, urg 0, length 0
14:44:01.240861 IP 192.168.2.230.40127 > 192.168.2.113.epmap: Flags [FPU], seq 3429184490, win 1024, urg 0, length 0
14:44:01.240862 IP 192.168.2.230.40127 > 192.168.2.113.telnet: Flags [FPU], seq 3429184490, win 1024, urg 0, length 0
14:44:01.240863 IP 192.168.2.230.40127 > 192.168.2.113.sunrpc: Flags [FPU], seq 3429184490, win 1024, urg 0, length 0
14:44:01.240863 IP 192.168.2.230.40127 > 192.168.2.113.pop3: Flags [FPU], seq 3429184490, win 1024, urg 0, length 0
tcpdump: Unable to write output: Broken pipe
time,ip_version,ip_ihl,ip_tos,ip_len,ip_id,ip_flags,ip_frag,ip_ttl,ip_proto,ip_chksum,ip_src,ip_dst,ip_options,tcp_sport,tcp_dport,tcp_seq
,tcp_ack,tcp_dataofs,tcp_reserved,tcp_flags,tcp_window,tcp_chksum,tcp_urgptr,tcp_options,payload
1642513441.240860,4,5,0,40,51462,,0,56,6,13090,192.168.2.230,192.168.2.113,[],40127,113,3429184490,0,5,0,FPU,1024,34707,0,[],000000000000
1642513441.240861,4,5,0,40,58227,,0,53,6,7093,192.168.2.230,192.168.2.113,[],40127,135,3429184490,0,5,0,FPU,1024,34685,0,[],000000000000
1642513441.240862,4,5,0,40,35227,,0,39,6,33677,192.168.2.230,192.168.2.113,[],40127,23,3429184490,0,5,0,FPU,1024,34797,0,[],000000000000
1642513441.240863,4,5,0,40,6686,,0,39,6,62218,192.168.2.230,192.168.2.113,[],40127,111,3429184490,0,5,0,FPU,1024,34709,0,[],000000000000
1642513441.240863,4,5,0,40,56935,,0,57,6,7361,192.168.2.230,192.168.2.113,[],40127,110,3429184490,0,5,0,FPU,1024,34710,0,[],000000000000
```

Fig. 6. Capture of comparison of raw packet capture and parsed CSV output.

## V. Preliminary results

Central components from this research are the automated scanner, task population, task monitor tool, task managing tool, packet capture retrieval tool, and packet capture parser. Each of these components has its own feature.

The automated scanner reads through the task list, determines available workers, and issues tasks to the representative worker for dumping packet captures meanwhile running a scan against the worker. In detail descriptions is found in section III-B.

Developed code using Python to iterate through given directories to identify parsable packet capture files. This parser outputs the relevant data into three separate files, each segmented into the protocol, details described in section IV.

Other smaller components were created during this research for the purpose of monitoring, maintenance, task management, worker preparations, and debugging. The task processor monitors ongoing tasks using the status in the task list and process checking. Monitoring processes were important during debugging, and ongoing scans and is one useful component for checking status for users using the component. During the population of tasks, a useful component to use is the task populator which iterates through the task list and adds an incrementing number to clone a given task in the task list to uniquely identify a task. Lastly, a retrieval component was developed to easily retrieve packet captures from each worker automatically. All these components are further described in section III-B.

The preliminary results are the development of code, and the creation of a virtual environment for generating amounts of synthetic packet captures in a cost-effective way. This means more tasks could be applied to the lab for either other tasks using Nmap or introducing other scanners. During the development, scalability was a priority to be able to increase the number of workers and introduction of other network scanners. During the conducted scans, packet captures for the modes aggressive, insane, polite, and normal were set to a limit of 100. These tasks have been issued two times, the first time with too much background noise, which needed to be eliminated fully. From here, a platform walkthrough eliminating such noise was conducted.

The packet captures were generated in the following time-frames seen in table I.

TABLE I
Packet capture time frames and counts

| Amount | Started | Finished | Timing template |
|---|---|---|---|
| 100 | Jan 18 02:57 | Jan 18 10:38 | aggressive |
| 100 | Jan 18 02:24 | Jan 18 11:05 | insane |
| 100 | Jan 18 11:05 | Jan 18 14:44 | normal |
| 100 | Jan 18 14:51 | Jan 18 23:22 | polite |
| 100 | Jan 18 04:02 | Jan 18 08:45 | sneaky |
| 10 | Jan 18 04:01 | Ongoing Jan 20 22:08 | paranoid |

## VI. Conclusion and further work

This research shows the importance of generating quality synthetic data for the use of a comparative analysis where the content is familiar when it comes to the analysis. If there exist patterns able to grow doubt of what the contents really are, the data are not of good quality for conducting a comparative analysis to identify various types of scans in a later stage. The importance of filtering away noise traffic not relevant for the analysis to generate a high quality is experienced. Having high-quality data early in the stage of developing a signature for given traffic results in a high-quality signature.

Other key aspects are the availability of a lab environment to generate quality packet captures and a worker setup that could be used for other tasks than only scanning conducted in this research. A fully automated setup enables, in certain scenarios, further tasking and an increased amount of packet generation.

Further work within the research is to conduct an analysis of the generated and processed packet captures. For the analysis, various Python modules have been researched to determine the way ahead and how to be able to identify patterns for each various scan. Through an analysis, patterns and other relevant packet characteristics could be extracted. In this research, the parser has converted packet capture files to comma-separated value files, which with the use of the Python library, pandas could structure a specific scan type with a given timing template into a larger data frame for analysis purposes. Matplotlib could be used to plot data into visualized compiled data to extract the most interesting findings.

After an analysis, the creation of IDS signatures is a planned step further in the process to output a signature as a final goal of the research.

## References

[1] Lockheed Martin, "Gaining the advantage," Lockheed Martin, Tech. Rep., 2015. [Online]. Available: https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/Gaining_the_Advantage_Cyber_Kill_Chain.pdf

[2] E. M. Hutchins, M. J. Cloppert, R. M. Amin *et al.*, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *Leading Issues in Information Warfare & Security Research*, vol. 1, no. 1, p. 90, 2011.

[3] W. Acosta, "Large-scale analysis of continuous data in cyber-warfare threat detection," in *The Proceedings of the 6th International Conference on Information Warfare and Security*, 2011, p. 317.

[4] R. J. Barnett and B. Irwin, “Towards a taxonomy of network scanning techniques,” in *Proceedings of the 2008 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries: Riding the Wave of Technology*, ser. SAICSIT ’08. New York, NY, USA: Association for Computing Machinery, 2008, p. 1–7. [Online]. Available: https://doi.org/10.1145/1456659.1456660

[5] A. Houmz, G. Mezzour, K. Zkik, M. Ghogho, and H. Benbrahim, “Detecting the impact of software vulnerability on attacks: A case study of network telescope scans,” *Journal of Network and Computer Applications*, vol. 195, p. 103230, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804521002290

[6] B. Pinkard and A. Orebaugh, “Nmap in the enterprise: your guide to network scanning,” 2008.

[7] M. Dabbagh, A. J. Ghandour, K. Fawaz, W. El Hajj, and H. Hajj, “Slow port scanning detection,” in *2011 7th International Conference on Information Assurance and Security (IAS)*. IEEE, 2011, pp. 228–233.

[8] R. Fekolkin, “Intrusion detection & prevention system: overview of snort & suricata,” *Internet Security, A7011N, Lulea University of Technology*, pp. 1–4, 2015.

[9] J. Liu and K. Fukuda, “Towards a taxonomy of darknet traffic,” *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 37–43, 2014.

[10] J. Mazel, R. Fontugne, and K. Fukuda, “A taxonomy of anomalies in backbone network traffic,” in *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2014, pp. 30–36.

[11] J. Liu and K. Fukuda, “An evaluation of darknet traffic taxonomy,” *Journal of Information Processing*, vol. 26, pp. 148–157, 2018.

[12] Z. Jammes and M. Papadaki, “Snort ids ability to detect nmap and metasploit framework evasion techniques,” *Adv. Commun. Comput. Netw. Secur*, vol. 10, p. 104, 2013.

[13] J. Arkko, M. Cotton, and L. Vegoda, “IPv4 Address Blocks Reserved for Documentation,” RFC 5737, Jan. 2010. [Online]. Available: https://rfc-editor.org/rfc/rfc5737.txt

[14] E. P. Gerich, “Guidelines for Management of IP Address Space,” RFC 1466, May 1993. [Online]. Available: https://rfc-editor.org/rfc/rfc1466.txt