Using Deep Packet Inspection in CyberTraffic Analysis

Luca Deri
ntop
Pisa, Italy
Email: deri@ntop.org

Francesco Fusco
IBM Research
Zûrich, Switzerland
Email: ffu@zurich.ibm.com

***Abstract*—In recent years we have observed an escalation of cybersecurity attacks, which are becoming more sophisticated and harder to detect as they use more advanced evasion techniques and encrypted communications. The research community has often proposed the use of machine learning techniques to overcome the limitations of traditional cybersecurity approaches based on rules and signatures, which are hard to maintain, require constant updates, and do not solve the problems of zero-day attacks. Unfortunately, machine learning is not the holy grail of cybersecurity: machine learning-based techniques are hard to develop due to the lack of annotated data, are often computationally intensive, they can be target of hard to detect adversarial attacks, and more importantly are often not able to provide explanations for the predicted outcomes. In this paper, we describe a novel approach to cybersecurity detection leveraging on the concept of security score. Our approach demonstrates that extracting signals via deep packet inspections paves the way for efficient detection using traffic analysis. This work has been validated against various traffic datasets containing network attacks, showing that it can effectively detect network threats without the complexity of machine learning-based solutions.**

Keywords—Deep packet inspection, Encrypted traffic analysis, Open-source.

I. INTRODUCTION

In the last decade, the Internet became ubiquitous and pervasive with millions of people connected to it all the time via smartphones and millions of IoT devices, both for industrial customers and residential customers. Our dependency on Internet services combined with the availability of always-on and high-bandwidth Internet connections offered new opportunities for cybercriminals. Modern cybersecurity criminals belong to well organised and well founded organisations as an economical reward for cyber crimes are constantly growing. Not surprisingly the level of sophistication of current cybersecurity attacks is constantly increasing and it is causing an arms race between attackers and responders. Timely detection of cybersecurity attacks is fundamental to protect organisations and their data from intruders. Detecting complex attacks such as Advanced Persistent Threats (APT) is extremely challenging and it usually requires dedicated teams of security experts who use their experience to identify suspicious network activities. Traditional intrusion detection systems, which are usually based on signatures and rule-based approaches, have shown their limitations in detection capability, especially when attackers and the infected hosts heavily rely on encryption to obfuscate communications and in all the cases when new threats appear for the first time. In order to improve upon rules and signatures, machine learning approaches are currently used in existing products to overcome those limitations. Machine learning promises to improve the detection capabilities for previously unseen threats, such as zero-day attacks, and also to reduce the problem of maintaining and updating a set of rules and fingerprints. While we do believe that machine learning technologies are playing and will play in the future an important role in cybersecurity, we strongly believe that domain knowledge and feature engineering have tremendous value for any detection problem.

In this paper, we highlight that despite the always increasing adoption of encryption technologies, deep packet inspection can still be used to extract very strong signals from the raw traffic. While one could feed those signals to machine learning based detectors, we highlight that when strong signals are available, one can greatly profit from them even with less sophisticated data processing technologies. By manually inspecting a large collection of malware traces, we observed that many suspicious activities related to malware can be easily detected in malware traffic, even when encrypted. We have implemented a system that detects many of those activities in real-time using deep packet inspection and exploits them to associate a reputation score to local and remote hosts.

We highlight that while the system still requires further validation in the field, our approach raises the bar for cyberattack detections for multiple reasons. Compared to machine learning systems, where the outcomes are usually hard to explain, our approach gives operators clear explanations of the reasons why a system can be considered suspicious. Additionally, it is efficient and does not require annotated data.

II. CYBERSECURITY AS A PROCESS

A. Motivation

Network traffic has changed dramatically in the past decade:

- Today most local and Internet network traffic is encrypted [1] not only because more recent protocols are using encryption by default, but also because traditional protocols that are as old as the Internet have been modified to support encryption [2].
- Hardening features typical of mobile devices [3] are deployed in desktop operating systems, making computer interactions more secure.
- The increased adoption of cloud-based services has drastically changed the communication patterns within enterprise networks. While more and more services such as email, backups, and name resolution are moving to cloud-based offerings, some north-south traffic communications are replacing former east-west communications. Therefore, on the Internet link, it is possible to observe traffic (mostly encrypted) that used to be exchanged in LAN in unencrypted form.
- Cloud-based IoT devices such as Google and Alexa-controlled devices have access to privileged physical resources (e.g. a gate, a power plug, or a video camera) and

need to be protected in order to avoid that breaches could expose sensitive information to intruders.

From the network traffic standpoint, these changes in communication patterns have an impact on network security tools as they have to deal with the reality that encrypted traffic is replacing Internet protocols that were traditionally simple to observe and monitor. While the transition towards new encrypted communications protocols happens at a high pace, popular open-source Intrusion Detection Systems (IDS) such as Suricata and Snort are still using outdated techniques to detect intrusions, which are at the same time error-prone and computationally intensive.

```
alert tcp any any <> any 443
(msg:"APT.Backdoor.MSIL.SUNBURST"; content:"|
16 03|"; depth:2; content:"|55 04 03|";
distance:0; content:"digitalcollege.org";
within:50; sid:77600846; rev:1;)
```

1. Example of a Snort/Suricata rule for detecting Sunburst Malware

Above we report a simple rule[1] for detecting a popular malware. This rule is used to detect TLS (Transport Layer Security) communications towards domain name digitalcollege.org. Instead of specifying something like "proto=TLS and SNI=digitalcollege.org" this rule searches for specific patterns in the packet payload (0x16 0x03 is the beginning of a TLS encryption block) at specific offsets that make it inefficient and subject to false positives. This is an example of limitations of popular tools that have been designed more than a decade ago where most traffic was exchanged in cleartext, IP addresses and service ports were relatively static (i.e. TCP/80 means HTTP, and no HTTP traffic on any other port different than 80) with no indirect cloud-based communications that make IP-address based rule ineffective as in the above example.

This work builds upon our practical security-oriented network traffic monitoring experience. The goal of this paper is to describe what are the techniques and methods that have been developed and deployed to effectively detect network attacks, and report suspicious communication patterns. Learned lessons, metrics, and methods have been implemented in the open-source tools nDPI and companion tools that have been made available to the Internet community for further extending this research work and providing practical tools to use in everyday life.

The main contribution of this work is to show how to implement robust traffic classification methods that allow cybersecurity threats to be identified. As described later in this paper, authors have not decided to follow a generic approach based on a plethora of features selected through a machine learning algorithm [6], but instead from traffic observation, they have created specialised metrics and simple statistical based models that enable behaviour modeling with ease thus reducing overall application complexity. This enabled the creation of fine-grained network models simple enough to be deployed with limited computational costs and able to fit into small processing units as those that can be found on modern network devices, this to implement pervasive security that would not be possible with complex traditional models.

B. *From Cleartext to Deep Packet Inspection (DPI)*

The migration from cleartext to encrypted communications and dynamic port usage has been a big change for many security tools as they were layered on the principle "one port, one protocol" (e.g. TCP/443 means TLS). Popular 2020's malware and trojans such as Dridex, Trickbot, and Emotet [4], use non-standard protocol ports to avoid IDS and thus hiding infection traffic. They leverage malspam (Malware Spam) or phishing to trigger the download of the trojan that then spreads inside the network to harvest information that is then sent back to the attackers on remote servers. The need to inspect and identify traffic regardless of the port being used, has been the main motivation to develop traffic inspection techniques based on DPI. As DPI toolkits are expensive, closed-source, and thus with limited extensibility, we have decided to create our own open source DPI toolkit named *nDPI* where we have implemented the traffic inspection techniques described later in this paper. The use of DPI in cybersecurity has several advantages as it conceptually splits the development of monitoring applications into two main blocks. The DPI layer is responsible for dissecting traffic and extracting metadata such as the HTTP URL or the TLS certificate, which are used to characterise traffic regardless of the IP and port the traffic is flowing to. The traffic analysis layer that sits on top of DPI can be simplified as filtering rules as those shown in Fig. 1 can be rewritten in a simpler format as already discussed, and also because this block does not have to deal with low-level protocol details but instead implement the business logic of the security application.

nDPI is not just yet another DPI toolkit capable of dissecting application protocols and extracting the corresponding metadata. nDPI can also extract from encrypted traffic several industry-standard traffic markers for encrypted communication. A popular client/server fingerprinting method is JA3 [5] (hence the variant names JA3C and JA3S), which is based on cryptographic information exchanged in ClientHello and ServerHello TLS packets. JA3, and its SSH counterpart named HASSH, is used by most cybersecurity applications to fingerprint the security toolkit used by a certain tool (e.g. OpenSSL vs. LibreSSL). Fingerprints extracted from JA3 and HASSH can help identify specific communications, but unfortunately, one can still observe false positives. For this reason in addition to industry-standard fingerprints, we have introduced in nDPI additional methods to uniquely fingerprint encrypted traffic as described in the following section.

C. *Fingerprinting Encrypted Traffic*

A great difference in methodology when analysing encrypted traffic with respect to clear-text is the inability to inspect the payload content and thus to characterise the nature of the communication. This means for instance that it is not possible to see if a cloud-initiated command requests our smart lamp to be turned on or off. For this reason, the best approach that can be adopted is to develop a reliable method for identifying similar traffic streams produced by the same (malware) application. This is to reduce the problem of detecting encrypted traffic streams to a malware signature. This is still the main approach used by popular IDSs despite intrinsic limitations of this methodology such as being unable to detect new threats and zero-days. Cisco Joy [7] has

[1] https://raw.githubusercontent.com/fireeye/sunburst_countermeasures/main/all-snort.rules

pioneered encrypted traffic fingerprinting by using a model making use of more than hundreds of generic traffic features including SPLT (Sequence of Packet Length and Times), bytes distribution (that is a Markov chain of bytes within a flow), and IDP (Initial Data Packet) that is a fingerprint of the initial flow packets.

III. From Fingerprinting to Lightweight Modelling

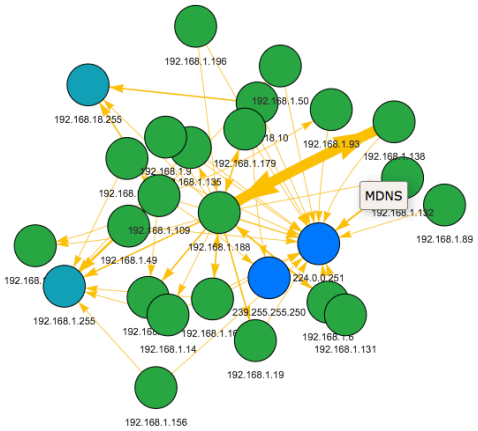
As already discussed, signature-based approaches as the one listed above, are only effective when large fingerprint databases are maintained and continuously updated. Therefore, signature-based approaches are only effective when there are large companies such as Cisco who are willing to dedicate entire teams to fingerprint suspicious traffic. In our case, we have decided to follow a different path. Instead of fingerprinting network traffic and comparing it against a database of known malware fingerprints, we have built a lightweight traffic model for each monitored host and continuously compared the observed traffic against the model. There are two families of devices we model:

- Single-purpose devices such as smart IoT devices, multimedia, printers, and NAS (Network Attached Storage) that carry on a well-defined task.
- General-purpose computers such as laptops and tablets that have not necessarily a predefined behavior.

For both families, we have created a model based on the observation of the device traffic for a period of time long enough to be able to map all possible communication flows. In our experiment, a day of observation is usually enough to grasp most communications but sometimes there are periodic weekly behaviours (e.g. a backup) that need to be added to an existing model. Such a model is based on three key observations all leveraging nDPI and described this section.

A. Service Map

East-west communications, i.e. interactions of the host with other hosts belonging to the same network.



2. Service map for intra-LAN communications

In this model, the nodes are local hosts and the arcs are the triplets <DPI protocol, source/destination IP/VLAN, destination port> where for DHCP-based networks we use the MAC instead of the IP address.

As we expect future communications to comply with this model, in case a new edge/node is added, a notice is triggered so that this model discrepancy can be analysed by a security analyst. In case of a positive decision about the discrepancy, the model can be extended with this new edge/node, otherwise an alert is triggered. Each edge can also be characterised by additional properties such as the flow creation frequency or the service name in case of encrypted communications, to enforce the checks of live traffic when compared to the existing model. The following section will report how the service map simplifies the detection of lateral movement caused by malware or unauthorised monitoring applications. According to our experience, the service map is much more effective yet simpler than other more sophisticated flow-based techniques.

B. Periodicity Map

Malicious applications often use beaconing techniques [8, 10] to connect with peers. Beacon messages are usually periodic communications. Being able to detect beacon traffic is a fundamental step to reveal the presence of suspicious communications. Detecting periodic communications is helpful in particular with low-volume conversations that can be easily hidden inside the overall traffic. Instead of using complex AI techniques [9] we have developed a simple algorithm for detecting periodic communications. Whenever a new flow is detected, we keep track of the quadruplet <source/destination IP, destination port, layer 4 protocol> and store it on a hash table. If a new flow with the same quadruplet is observed, we start checking the periodicity. Entries idle for too long (currently we limit our analysis to one hour periodicity to put an upper bound to resource utilisation) are periodically purged, as well entries that show a periodicity drift greater than double the reported periodicity. For instance, if the system detected a periodicity of 60 seconds, as soon as a periodicity measurement exceeded 120 seconds, the entry is marked as non-periodic and discarded. Quadruplet updates observed with less frequency not exceeding one second are ignored as we consider these flows as part of the same communication, a typical event for many protocols such as HTTP for instance. Combining periodicity with DPI enabled us to quickly identify malicious beaconing by checking the application protocol bound to this periodicity. For instance, periodic communications using unknown (i.e. a protocol that was not recognised by the DPI engine) or potentially malicious (e.g. IRC) protocols can be used to immediately raise an alarm.

C. Security Risks

As previously discussed, nDPI has been extended to report more than the application protocol and metadata information typical of DPI libraries. In particular, every detected protocol is classified into categories (e.g. file transfer but also mining and banned sites just to name a few) based on static protocol knowledge (e.g. Facebook is classified as a social network) and lists freely available on the Internet (e.g. EmergingThreats provides a constantly updated list of malware sites). In addition to all this, nDPI implements the concept of *security risk*, that is a score assigned to a DPI-detected flow that indicates how malicious is such communication based on the security issues that have been detected. The list of supported risks has been created by analysing several network traces publicly available on the Internet[2] to spot the most popular techniques used by

[2] In particular in this work we have used many traces available at https://www.malware-traffic-analysis.net and https://www.netresec.com/.

malware applications. To date nDPI supports the following security risks:

- Possible XSS (Cross Side Scripting) attack, SQL Injection, RCE (Remote Code Execution) Attempt.
- Binary application transfer (e.g. when downloading or uploading executable applications).
- Known protocol on a non-standard port (e.g. when using TLS on a port other than 443).
- TLS and QUIC: obsolete protocol version, weak cipher, certificate expired/mismatch/self-signed, TLS not transporting HTTPS (this is detected using the ALPN TLS extension) and TLS without SNI (Server Name Indication) that definitively indicates something wrong with this communication. Additionally for TLS nDPI detected the usage of ESNI (Encrypted SNI) which is a potential method for hiding the server name a client is connecting to.
- HTTP: suspicious user agent, numeric IP host, suspicious URL, or protocol header. Furthermore, nDPI reports about suspicious content that is triggered whenever the exchanged data is not compliant with the advertised Content-Type header. Many malware [11] in particular when exfiltrating data towards a remote site, perform HTTP POST using Content-Types such as text/plain or x-www-form-urlencoded but the exported data is in binary format.
- Suspicious DGA (Domain Generated Algorithm) domain name. This is implemented using a lightweight bigram-based technique inside the library.
- Malformed protocol traffic (e.g. DNS packets jeopardised to implement data transfer such as a VPN-over-DNS).
- SSH: obsolete client/server application version or weak/obsolete cipher. Contrary to TLS, in SSH the list of popular implementations is rather short and nDPI includes a list of popular SSH implementation versions that can be checked for obsolescence.
- SMB (Server Message Block, popular protocol used mostly on Windows-based networks for sharing data across systems) insecure version.
- Unsafe protocol: used when a detected protocol is either insecure (e.g. POP3 is unsafe as it exists a corresponding encrypted protocol named POP3S), potentially dangerous (examples include Tor, Stealthnet, HotspotShield), or definitively dangerous (e.g. SMBv1) when detected on a network. In particular, SMBv1 is well known for being vulnerable to attacks and thus compulsory to spot and upgrade or isolate hosts using it.

The concept of security risk is defined at the level of individual flow, and it does not take into account other higher-level indicators such as host's reputation or behaviour. In this work, we rely on the concept of *reputation score*, which can be applied to hosts, subnets, autonomous systems, countries, etc. For this reason, we use external trusted feeds (e.g. IP blacklist maintained by Emerging Threats) to complement network traffic knowledge with host reputation. This allowed us to implement additional security risks whenever we observe network communications with hosts present in blacklists.

D. Mapping Security Risks to Score

As already discussed most IDSs (e.g. Suricata and Zeek) report information at flow level, just as nDPI does, but without using DPI methods to detect the application protocol. This approach is a good foundation for detecting suspicious behaviour but it needs to be consolidated in a few dimensions:

- Time: is the detected behaviour steady or it changes over time? Strong changes in behaviour can indicate that something suddenly changed as it happens when a resource is under attack.
- Resource: a flow score cannot have a static severity but it needs to be interpreted within a context. For instance, suppose the system detected a client that connects to a server over TLS issuing requests with weak ciphers and no SNI. This severity associated with this fact cannot be uniform, but it must be higher on the client that performed the action, and less severe on the server that just received the request. On the other hand, on a flow where a client contacts a server using an expired TLS certificate, the latter should have a higher severity with respect to the client.

In order to add weight to detected risks and consolidate them, we introduce the concept of score. The *risk score* is a numerical value which indicates how suspicious a behavior is. High score values corresponds to high risk, zero means no risk. For every flow we define:

- Flow score: a numerical indicator that indicates the risk associated with the flow. It is computed as the sum of all the individual security risks detected for the flow.
- Client and server score: numerical indicator used to bind the individual security risks to flow client/server. As already discussed earlier in the section, the flow score is split between the client/server according to the severity associated with the attacker/victim detected risk.

The score is updated every minute and corresponds to the sum of the individual active flow scores of the host. Within a minute, a host with no invalid behaviour reports has a zero score, which instead increments as flows with a non-null flow risk and active in such a minute are observed. This way the score is a numerical value that can increase and decrease over time, and that indicates how bad is the detected host behaviour within such a minute. It is possible to consolidate the score at a higher level such as subnet or autonomous system, by simply summing the observed client/server score according to all the hosts belonging to such subnet and autonomous system. This is a simple yet effective technique to consolidate threats at a higher granularity than individual hosts with an absolute (i.e. not normalised) value indicating the risk of groups of hosts in the network.

As discussed in the following section, the score is effective as long as invalid signals are ignored. For example, it is unfortunately not uncommon in IoT setups to keep in networks outdated or insecure devices which are impossible to upgrade or replace. Those devices are usually deployed in hardened networks to isolate the main networks from them. In this case, the flow risk for these IoT devices must be interpreted with a correction factor that assigns a lower severity for known issues or completely ignores such well-known issues at all. This correction step is performed as soon as the monitoring system is deployed: it can be very specific (e.g. ignore specific risks

for specific devices) or coarse (e.g. don't consider any risk produced by host X). In our experience, network administrators often prefer to ignore specific hosts as this is simpler to configure instead of selectively ignoring specific risk/host combinations, even though this is not a practice we encourage as the system is completely blind with respect to security threats affecting those devices that have been put on a whitelist.

IV. VALIDATION

This work has been validated by using it in real traffic scenarios including the network of a large University campus and several business networks, not to mention the improvements and testing done by the open source community that used it in various heterogeneous networks. The goal of this section is to show how the tools described in this paper have been used to successfully identify malware traffic and effectively spot the attacker and the attacked hosts. Unfortunately due to privacy constraints and regulations we are not allowed to make this traffic available to the research community. Therefore, in this section, we will report only the results of some experiments based on publicly accessible datasets that can be used to reproduce the findings reported in this paper.

A. *Trickbot Analysis*

Trickbot is a popular banking trojan and malware targeting Windows systems. Using nDPI toolkit we have analysed a trickbot trace[3] that is about one hour long. Out of the 279 flows contained in the trace, nDPI has identified 242 flows with risk including: obsolete TLS protocol, known protocol on non standard port, HTTP requests with numeric IP address, and self-signed TLS certificate. Furthermore three additional flows are even more suspicious as nDPI has detected a binary application transfer that is when trickbot infects other computers. As previously explained, we have assigned a numerical score to each of the nDPI risks based on the severity of the identified issue: for instance a binary application transfer in HTTP has a high score when compared to obsolete TLS version. The flow risks have been complemented with additional indicators that are based on blacklists such as JA3 fingerprint blacklist (https://sslbl.abuse.ch/blacklist/ja3_fingerprints.csv) and snort IP blacklist (https://snort.org/documents/ip-blacklist). Accounting the reputation score based on the flow risks, makes it evident that host 10.9.25.10 is the infected host as it accounts a client score of 7,940. Using data binning techniques it is possible to further expand our analysis and better fingerprint this traffic malware. Whenever a compromised host has been identified, it is compulsory to see if in the network there are other similar traffic flows that have not yet been marked as infected. Below you can find an example of TLS traffic from the infected hosts that shows how nDPI fingerprints the traffic.

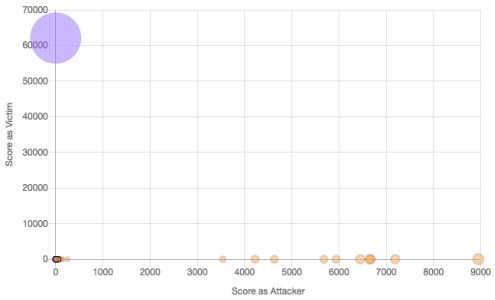
```
10.9.25.101:49469 <-> 5.53.125.13:447|3,1,1,1,1,1|
8,0,0,0,0,0|2.406|6734f37431670b3ab4292b8f60f29984|
623de93db17d313345d7ea481e7443cf|DD:EB:4A:36:6A:2B:50:DA:
5F:B5:DB:07:55:9A:92:B0:A3:52:5C:AD

10.9.25.101:49482 <-> 185.90.61.116:447|3,1,1,1,1,1|
8,0,0,0,0,0|2.406|6734f37431670b3ab4292b8f60f29984|
623de93db17d313345d7ea481e7443cf|DD:EB:4A:36:6A:2B:50:DA:
5F:B5:DB:07:55:9A:92:B0:A3:52:5C:AD

10.9.25.101:49498 <-> 195.123.221.104:447|3,1,1,1,1,1|
8,0,0,0,0,0|2.406|6734f37431670b3ab4292b8f60f29984|
623de93db17d313345d7ea481e7443cf|DD:EB:4A:36:6A:2B:50:DA:
5F:B5:DB:07:55:9A:92:B0:A3:52:5C:AD
```

3. nDPI's Trickbot flow analysis

The first column contains the flow IP and ports, the second one the packet length bins, then the packet inter-arrival bins, the bytes entropy, the JA3C/JA3S and finally the server certificate hash. As you can see they are identical flows all originated by the same infected host. The use of bins allowed us to create a fingerprint key, that solves the problem of uniquely identify trickbot flows, that was the limitation of JA3 due to the fact that, as already discussed, JA3 is affected by false positives. Furthermore the unique tuple JA3S/certificate fingerprint leads us to the conclusion that the above servers are part of the trickbot network. This technique is effective also with other datasets as described in the following section. The score based on security risks allowed us to easily cluster hosts that are under attack.

B. *CSE-CIC-IDS2018*

The above experiment has demonstrated how the use of score allows attackers/attacked hosts to be identified. Being the experiment based on a one hour long trace, it has not taken into account all the techniques described in this work such as the service map. In this experiment we have analysed the popular CSE-CIC-IDS2018 dataset[4] that includes several traces in total about 10 days long. In [12] the dataset has been analysed using a complex machine-learning based system, and in this experiment we want to position this work against it. As this dataset has been annotated by the authors, we have the ability to check the reported results; due to space constraints we report results for one day of traffic in the following picture..



4. CSE-CIC-IDS2018: Tuesday-20-02-2018 Traffic Analysis

The bubble chart above depicts on the X axis the host client score, and on the Y axis the host server score. Each bubble represents a host whose size is the host score: the larger is the bubble, the highest is the host score. As you can see the attacker and victim hosts can be easily spotted as they are far away from origin where most hosts reside. This result matches exactly (no false positives or negatives) the expected results reported in the dataset annotation using a much simpler approach that does not requires hundred of features or an annotated dataset for training a model.

[3] The trace is available in pcap format at https://www.malware-traffic-analysis.net/2019/09/25/

[4] The dataset is freely available at https://www.unb.ca/cic/datasets/ids-2018.html

C. nDPI Performance and Memory Evaluation

nDPI analyses the first few connection packets in order to detect the application protocol and perform flow analysis. For UDP-based protocols such as DNS two packets (one for the request and one for the response) are sufficient, whereas for TCP-based protocols the number of packets depends on the protocol ranging from a minimum of 6 packets for HTTP to about 13 for TLS 1.2 where the protocol negotiation phase is longer due to the TLS hello and certificate exchange. As DPI is involved only during the initial flow analysis, its overhead is proportional to the number of flows, and, only for the first few packets. During protocol detection, nDPI needs to allocate a 2.4 KB temporary datastructure used to store temporary flow information until the dissection is completed producing metadata and reporting the application protocol. This means that both the memory and CPU overhead is accounted only for the first flow packets as once dissection is over, nDPI flow memory can be freed and no further nDPI traffic processing is necessary. Application protocol dissectors are sequentially run by nDPI based on the matching probability (e.g. for TCP/80 the first dissector to test is the one for HTTP and non-TCP dissectors are immediately discarded) until a match is found or no protocol matched. Hence the worst performance case is with synthetic traffic as no dissectors will match and nDPI will need to try all the potentially matching dissectors. In this case the ndpiReader tool can process per core ~40 Kpps with respect to ~1.8 Mpps that we have observed with real traffic captured on a public Internet link corresponding to about 10 Gbit/s.

D. Score Evaluation

The work presented in this paper has demonstrated to be effective and able to detect real-life threats without the need to train a model on the target network, hence to be able to potentially spot zero-day threats. For example, the recent attack on MS Exchange servers [13] could be spot by using the service map to detect unexpected movements of Exchange servers as well unveil new periodic communications towards remote locations using the periodicity map. However the concept of score can sometimes lead to false positives when traffic contains unexpected flows. Typical examples are flows that have a non zero score but that are not malicious (e.g. for some reason the systems cannot be upgraded and they have been protected with strict ACLs) including insecure IoT devices that perform potentially malicious operations over HTTP, or a local web server that uses a self-signed certificate. Our work is based on the principle that non-zero flow score indicates a problem, hence the above examples must be manually labeled as exceptions via rules. Such rules can be compiled by running our tool on the monitored network for some time (e.g. one day), looking at the reported alerts and configuring exceptions for flows that are not supposed to be used for computing the score. This approach gives network administrator the flexibility to deal with hardware devices or legacy applications that cannot be modified.

V. Conclusions

This paper has covered the design and implementation of a novel approach to cybersecurity based on deep packet inspection and traffic analysis methods. This work originated from the experience gained while developing open source traffic monitoring software: it has been applied to real traffic as well publicly available datasets. In this paper we have described an approach that is simple to implement and computationally efficient both in memory and processing time, as well effective in detecting modern malware, ransomware and trojans using encryption. The software developed in this research work has been released as open-source, enabling security researchers to reproduce our results and to build more advanced solutions on top of our extractors.

Code Availability

The nDPI source code is available under LGPLv3 license at https://github.com/ntop/nDPI. It includes all the traffic analysis algorithms, flow risk and encrypted traffic analysis described in this paper. The code that implements the risk score is part of ntopng released under GPLv3 and available at https://github.com/ntop/ntopng.

Acknowledgment

The authors would like to thank Braintrace Inc. that supported the development of nDPI and provided a playground for testing the techniques and algorithms described in this paper. We also would like to thank Alessio Perugini, and the ntop team for comments, suggestions, and evaluating the techniques, code and algorithms described in this paper.

References

1. M. Meeker, “Internet Trends 2019”, https://www.bondcap.com/report/itr19, Bond Inc, June 2019.
2. P. Hoffman, and P. McManus, “DNS Queries over HTTPS (DoH)", RFC 8484, October 2018.
3. Dawson, Maurice & Wright, Jorja & Omar, Marwan. (2016). “Mobile Devices: The Case for Cyber Security Hardened Systems”. In New Threats and Countermeasures in Digital Crime and Cyber Terrorism, IGI Global, 10.4018/978-1-4666-8751-6.ch047.
4. C. Patsakis, and A. Chrysanthou, “Analysing the fall 2020 Emotet Campaign”, Tchnical Report, University of Piraeus and Neurosoft, https://arxiv.org/pdf/2011.06479.pdf, November 2020.
5. J. Althouse, "TLS Fingerprinting with JA3 and JA3S”, https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967, Salesforce Engineering, 2019.
6. H. Hindy et al., “A Taxonomy of Network Threats and the Effect of Current Datasets on Intrusion Detection Systems”, IEEE Access (2020).
7. B. Anderson, D. McGrew, "Identifying encrypted malware traffic with contextual flow data”, Proceedings of the 2016 ACM workshop on Artificial Intelligence and Security, 2016.
8. M. Haffey, M. Arlitt, and C. Williamson. "Modeling, Analysis, and Characterization of Periodic Traffic on a Campus Edge Network.”, Proceedings of *2018 IEEE 26th International Symposium MASCOTS*. IEEE, 2018.
9. Y. Borchani, "Advanced malicious beaconing detection through AI." *Network Security* 2020.3 (2020): 8-14.
10. B. AsSadhan, J. M. F. Moura and D. Lapsley, "Periodic Behavior in Botnet Command and Control Channels Traffic," *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*, Honolulu, HI, 2009, pp. 1-6, doi: 10.1109/GLOCOM.2009.5426172.
11. H. Poston, “Network traffic analysis for IR (Incident Response): Content deobfuscation”, INFOSEC, https://resources.infosecinstitute.com/topic/network-traffic-analysis-for-ir-content-deobfuscation/, October 2019.
12. I. Sharafaldin, A. Habibi Lashkari, and A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”, Proceedings of *4th International Conference on Information Systems Security and Privacy (ICISSP)*, 2018.
13. T. Lee, I. Ahl, and D. Hanzlik, “Detecting and Defeating China Chopper Web Shell”, https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/rpt-china-chopper.pdf, FireEye Labs, 2021.