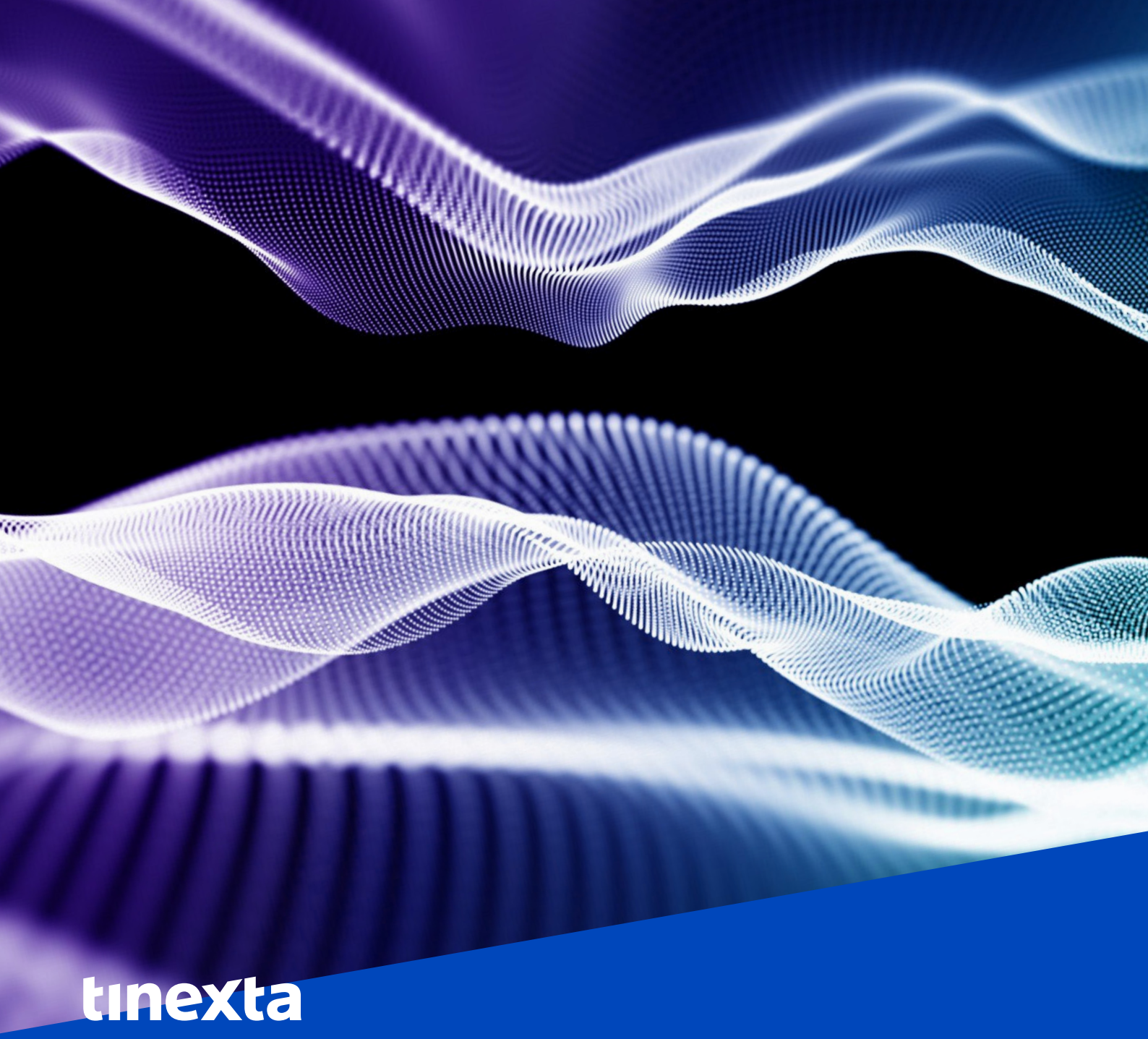


tinexta
defence

Open source release: NetFlowMeter

#TinextaDefenceBusiness

Malware Lab

Summary

Our Malware Lab

Tinexta Defence Malware Lab daily performs dissection of malware with the aim of timely understanding the technological evolutions of attacks, consolidating the knowledge of necessary to make more effective and faster the process of incidents responding, contributing to spreading information about emerging threats into the expert's community and among its clients.

Malware Lab analysts are continuously engaged in searching and experimenting new analysis tools, for increasing accuracy and scope of action with regard to the proliferation of new evasion and anti-analysis techniques adopted by malware.

The Malware Lab is also committed to the development of proprietary tools for malware analysis and supporting the management and response of incidents.

Besides malware analysis, Malware Lab ideated and implemented an automatic process of extraction of **Indicators of Compromise (IOC)** that is daily run on dozens of new malwares, intercepted in the wide for populating our Knowledge Base.

Corrado Aaron Visaggio

Group Chief Scientist Officer & Malware Lab Director

a.visaggio@defencetech.it

Autori del report:

- Ermes Pennucci: Cybersecurity Analyst

1. Open source release: NetFlowMeter

1.1 Introduction

Here in Tinexta Defece, we heavily invest in research and development on applications and tools for cybersecurity. One of our AI teams has recently been focused on the topic of network intrusion detection (IDS) using machine learning techniques. This field is very active in the research community and many datasets and tools have been developed in recent years. Open-source software is a strategic pillar of Tinexta Defence's core mission, fostering transparency, accelerating innovation, and strengthening our collaborative ecosystem, which is why we have officially launched our GitHub channel to share selected research outputs and engage with the wider community.

In this report we introduce one of the tools we have developed internally during said research activity which is now being released as open source: NetFlowMeter. NetFlowMeter is a flow analyzer for processing raw network traffic (pcap files) and extracting a set of features for use with machine learning.

It is now available on GitHub https://github.com/DefenceTechSecurity/NetFlowMeter under the MIT license.

NetFlowMeter shares part of its name with another tool called CICFlowMeter[1] which has been widely used in the research community for the same purpose. NetFlowMeter was born as a reimplementation of CICFlowMeter with the aim of improving its performance and fixing some bugs that were affecting the quality of the produced datasets.

1.2 State of the art of network flow analysis tools

One of the most influential datasets in this field is CICIDS2017[2], developed by the Canadian Institute for Cybersecurity (CIC). This dataset simulates a realistic network including a number of users performing normal activities as well as an attacker performing different types of attacks. This dataset has been widely used in academia for benchmarking machine learning algorithms for intrusion detection.

The dataset is freely available and includes both the raw pcap files as well the already processed CSV files containing the features extracted from the traffic flows. The features are extracted using the original CICFlowMeter, developed by the same team.

CICFlowMeter has been already independently evaluated in other studies which found a number of issues that affected the quality of the output. One of such studies is "Error Prevalence in NIDS datasets: A Case Study on CIC-IDS-2017 and CSE-CIC-IDS-2018"[3] where the authors found a number of labeling errors in CICIDS2017 as well as some bugs in CICFlowMeter that caused certain features to be incorrectly calculated. The paper came with a revised version of CICFlowMeter[4] fixing the identified bugs. Note that throughout this report we will only refer to this updated version of CICFlowMeter.

This is where our research team stepped in. After evaluating the available literature we started building automation pipelines to collect network traffic from internet-exposed machines and produce our own datasets.

1.3 NetFlowMeter

As we scaled up our data collection we started to observe performance issues with CICFlowMeter: pcap files larger than 400MB would cause it to become extremely slow and consume excessive resources during processing. Some issues could be mitigated by tweaking the Java Virtual Machine (JVM) parameters but overall the tool was not scaling well, this was being a major bottleneck in our workflow.

After investigating the codebase we decided that it would have been more efficient to start building our own tool from scratch rather than trying to fix and optimize the existing codebase. This is how NetFlowMeter was born.

Our initial goal was to reimplement CICFlowMeter in C#, a language that better integrates with our existing infrastructure, aiming to achieve output compatibility with the original tool. This allowed us to user our existing dataset to identify discrepancies in the output as well as to use NetFlowMeter as a drop-in replacement without having to immediately re-train our models.

The following is a comparison table of processing time and memory usage between CICFlowMeter and NetFlowMeter and the results speak for themselves:

PCAP Size	Tool	Time	Memory
500 MB	CICFlowMeter	2m 1s	3 GB
	NetFlowMeter	9s	550 MB
400 MB	CICFlowMeter	5m 40s	4 GB
	NetFlowMeter	13s	1 GB

In fact, on average NetFlowMeter is 10x faster and uses 5x less memory than CICFlowMeter. In the 400MB test case which is a particularly bad case for CICFlowMeter due to the GC overhead, NetFlowMeter is 26x faster. Note that maximum performance was not the goal, this is still single-threaded code and there is likely room for further optimizations.

Memory usage needs additional considerations: CICFlowMeter keeps all flows in memory until the end of the processing, this means that memory usage will keep increasing as more packets are seen. This does not scale well for extremely large pcap files or, more commonly, real time network analysis.

NetFlowMeter has to follow the same approach to be compatible but it also includes a command line option to periodically garbage collect closed flows, this allows to keep memory usage as low as 100MB while processing bigger datasets and real-time traffic in the future. The results obtained with this option however will not be compatible with CICFlowMeter.

1.4 Going forward

While developing NetFlowMeter we put together a test suite of pcap files captured from internet-exposed hosts and used it to continuously validate the output of our implementation against the original CICFlowMeter. This allowed us to identify a number of previously unknown bugs in CICFlowMeter which we documented in the code under TODO tasks.

In the future we plan to release a “2.0” version of NetFlowMeter which will break compatibility with CICFlowMeter in order to fix these issues and improve the quality of the produced datasets.

One example of such issues can be seen in the following code snippet.

```
// TODO: CICFlowMeter bug we imitate for output compatibility.
// This makes one of the header metrics always 0. This happens because in cic only the first packet
handler updates this value while the "normal" handler uses min(current, next) which for one of the
directions is always 0
// Remove the following two lines to fix the output.
if (isFirstPacket)
    otherMetrics.HeaderBytes.AddValue(0);
```

Here we have a specific metric that is only initialized in one “direction” of the flow and never updated in the other direction, meaning that for each flow only the server or the client side of the flow will have a non-zero value for this metric. We believe this is not intended and simply a logical bug in the initialization of the metric.

Our final compatibility result is 100% on our own curated dataset but only around 95% on CICIDS2017, meaning that in the general case there are still some discrepancies in the output. We manually compared the output of both tools and discovered a few additional bugs that we documented in the regression testing code. However, given the current roadmap we decided that exactly replicating their behavior is not going to be a priority.

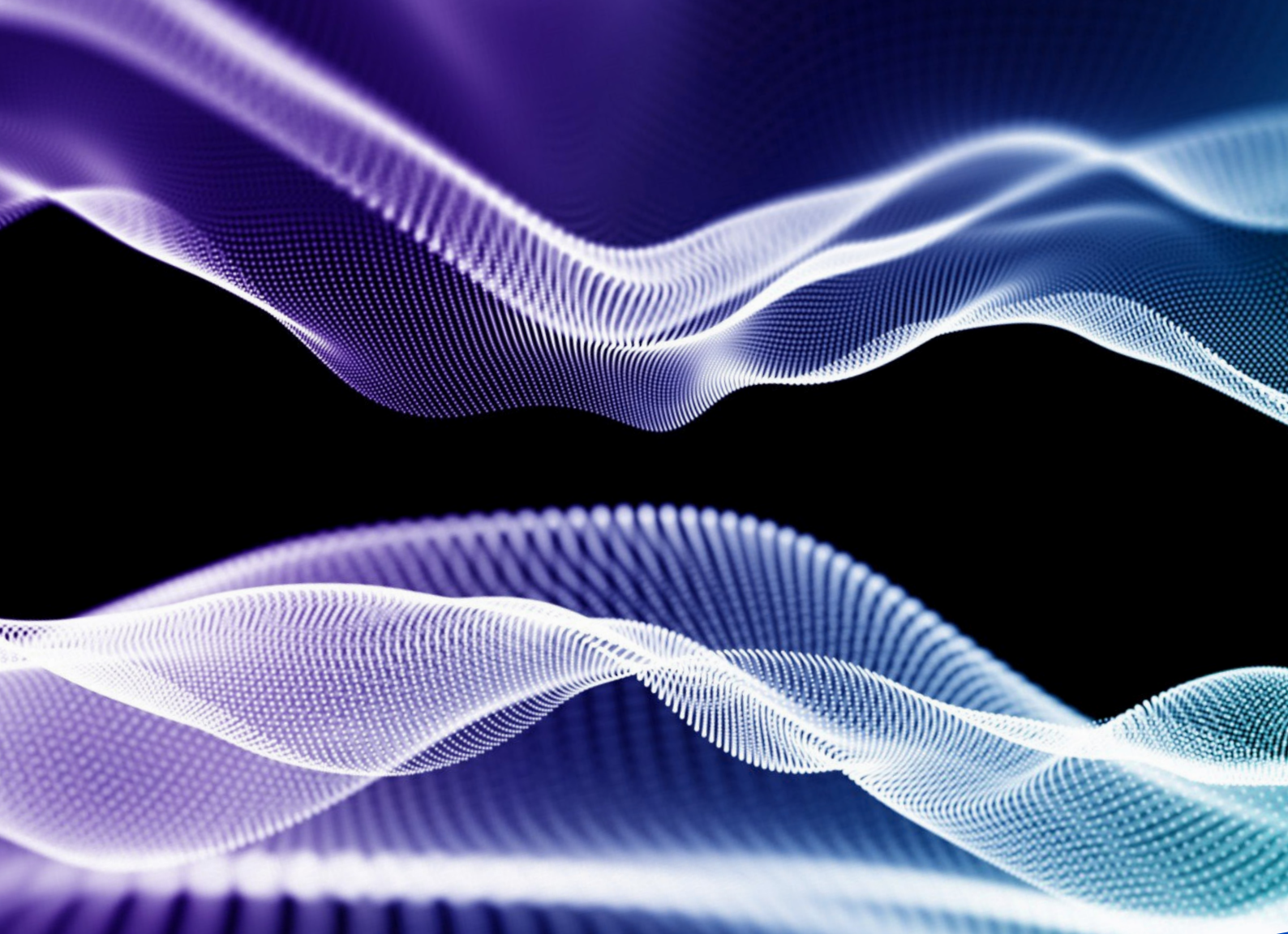
1.5 References

[1] https://github.com/ahlashkari/CICFlowMeter

[2] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018.

[3] L. Liu, G. Engelen, T. Lynar, D. Essam and W. Joosen, "Error Prevalence in NIDS datasets: A Case Study on CIC-IDS-2017 and CSE-CIC-IDS-2018," 2022 IEEE Conference on Communications and Network Security (CNS), Austin, TX, USA, 2022, pp. 254-262, doi: 10.1109/CNS56114.2022.9947235.

[4] https://github.com/GintsEngelen/CICFlowMeter



tinexta
defence

Next | Donexit | Foramil | Innodesi

Via Giacomo Peroni, 452 – 00131 Roma
tel. 06.45752720 – info@defencetech.it
www.tinextadefence.it

#TinextaDefenceBusiness