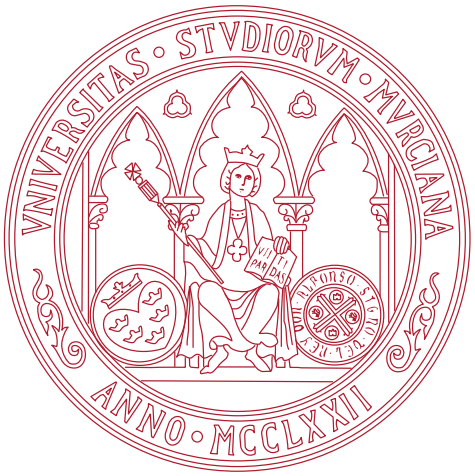


UNIVERSITY OF MURCIA

FACULTY OF COMPUTER SCIENCE

Enhancing DGA-Based Botnet Detection Beyond 5G with On-Edge Machine Learning

Aprendizaje Automático en el Edge para la Mejora de la Detección de Botnets DGA en Redes 5G y Futuras

Author

Mattia Zago

Thesis supervisors

Assoc. Prof. **Manuel Gil Pérez**, *Ph.D.*
Prof. **Gregorio Martínez Pérez**, *Ph.D.*

Murcia, June 2021

The following PhD Thesis is a compilation of the next published articles, being the PhD student the main author in all of them:

- Mattia Zago, Manuel Gil Pérez, and Gregorio Martínez Pérez. "**Scalable detection of botnets based on DGA: efficient feature discovery process in machine learning techniques**." *Soft Computing* 24 (2020): 5517-5537.
  DOI: 10.1007/s00500-018-03703-8
  J.I.F 2019: 3.050 (Q2)

- Mattia Zago, Manuel Gil Pérez, and Gregorio Martínez Pérez. "**UMUDGA: a dataset for profiling DGA-based botnet**." *Computers & Security* 92 (2020): 101719.
  DOI: 10.1016/J.COSE.2020.101719
  J.I.F 2019: 3.579 (Q2)

- Mattia Zago, Manuel Gil Pérez, and Gregorio Martínez Pérez. "**Early DGA-based botnet identification: pushing detection to the edges**." *Cluster Computing*, in press.
  DOI: 10.1007/s10586-020-03213-z
  J.I.F 2019: 3.458 (Q1)

Contents

Acknowledgements

Writing this dissertation marks the conclusive act of this Spanish arc of my life, which would have never been possible without the invaluable support of countless people. This document is their achievement as much as it is mine: my sincerest gratitude goes to all of you. Nonetheless, here follows in no particular order a collection of people that I wish to separately thank.

- To my family, for providing all the means and support that I needed.
- To Sabrina, for enduring all these years despite the distance.
- To Pantaleone, for being always there when I need him most.
- To the people in the research group, for making the university feel like a second home.
- To Gregorio and Manuel, for believing in me when nobody else did it.
- To the Erasmus Student Network volunteers, for teaching me the importance of balancing personal and professional lives.
- To Alexandra, Giorgia, Javier, Jorge, Lee, María, Paola, Sander, and Sergio for demonstrating that the borders are just a line on a map.
- To Stefano, Fabio, and Giuseppe, for being the best friends I could have hoped for.

Finally, this work has been supported by a predoctoral grant from the Spanish National Cybersecurity Institute (INCIBE) within the program "Ayudas para la Excelencia de los Equipos de Investigación Avanzada en Ciberseguridad" ("Grants for the Excellence of Advanced Cybersecurity Research Teams") with code `INCIBEI-2015-27353`.

Abstract

Notwithstanding the scientific community's efforts and results, malwares are still wreaking havoc of computer networks; among these threats, botnets are growing at an alarming rate and have been responsible for dangerous attacks. Indeed, in the past five years, notorious botnets such as Mirai, Roboto, or Kraken have been a primary target of the cybersecurity community. However, independently from the purposes of these malwares, the botnets are characterised by a common point of failure, *i.e.*, the communication channel. Infected devices need to reach out to the Command & Control (C&C) servers to download second-stage infections, perform malicious actions, or await further commands. As the infected devices are already connected to the internet, TCP/IP connections have been widely abused, notwithstanding the providers' efforts in blacklisting IPs and sinkholing fully qualified domain names (FQDNs). Domain generation algorithms (DGAs) have grown to a conventional approach to elude detection algorithms by generating pseudo-random rendezvous-points, *i.e.*, the C&C servers FQDNs. Although many machine learning (ML)-oriented frameworks have been theorised to identify and intercept DGAs, the problem is yet to be solved. As such, this PhD thesis's scope is to analyse the DGAs' outputs, known as algorithmically generated domains (AGDs), to provide a set of ML tools and privacy-aware methodologies that help identify these evasive patterns.

To be more precise, the objectives achieved throughout this research are twofold. On the one hand, this thesis aims to provide a characterisation of the DGAs aspects, including, among others, a comprehensive survey of previous literary contributions, data sources, and ML-based approaches for DGA-based botnet detection. On the other hand, it aims to integrate and improve the state-of-the-art by providing methods, strategies, and technologies to enable DGA-based botnet detection at scale. Specifically, signature patterns are identified in malicious AGDs using Natural Language Processing (NLP) techniques, and, the resulting learning models are designed as services to be dynamically deployed anywhere on the network.

As a result, this research encompasses literary survey, theory and framework crafting, experiments design and evaluations, and knowledge gaps identification and discussions. Under the compendium modality, the three chapters composing this PhD dissertation are outlined as follows.

- Firstly, a state-of-the-art survey on ML approaches to DGA-based botnet detection; the first chapter reports on supervised and unsupervised algorithms, their features sets, the definition of use cases and experiments, and, ultimately, the outline of

multiple research challenges to guide the thesis. Eventually, the experimental findings lay the foundations for AGDs formal and verifiable study.

- Secondly, a comparative analysis of the data sources to power ML frameworks; the second chapter reports on the published datasets by providing a formal comparison and discussion on multiple orthogonal properties. In the same article, the University of Murcia Domain Generation Algorithm Dataset (`UMUDGA`) is introduced as the most complete, balanced, and up-to-date collection of DGA-related data, featuring 50 malware classes for a total of 30+ million FQDNs. Eventually, the exploratory analysis reported in the article suggests that ML solutions to precisely pinpoint the malware variant based on AGDs pattern recognition are feasible.

- Thirdly, a proof-of-concept framework where the detection of DGA-based botnets is deployed as a security service on edge; the third chapter compares and examines architectural edge intelligence (EI) approaches to enable scalable detection in fifth generation (5G) networks and beyond. In the article, the experimental evaluation demonstrates that AGD detection is not only reasonable and achievable, but it is also plausible to expect to have deployed such detection capabilities on the networks' edges and eventually on the user equipments (UEs).

In summary, the chapters composing this PhD dissertation promote cohesive research exploring, analysing, and ultimately tackling the DGA-based botnets. Following this Ariadne's thread, each chapter is self-contained and provides critical insights on the research challenges from a different perspective; together, these contributions depict a clear description of the research niche summarised in the thesis. However, although conclusive on the explored subjects, some questions mooted by this research remain unsolved. Prime among them is whether it will be feasible to provide anonymous, exchangeable, and trustworthy profiles of AGDs to enable collaborative and federated detection models without harming users' privacy.

Resumen

Ahora que "todo" se ofrece como servicio, maximizar el rendimiento y minimizar la latencia son requisitos fundamentales de cualquier proyecto, ya sea éste de investigación y desarrollo o comercial. Esto es especialmente relevante considerando los millones de dispositivos que van a estar conectados gracias a las redes de quinta generación (5G), y que solo representan una parte de los que se esperan que lleguen a estar conectados a las redes futuras (del inglés, Beyond 5G o B5G). Sin embargo, en este entorno la ciberseguridad es una necesidad que sigue estando infravalorada: no hay normas de obligado cumplimiento en los estándares, en particular cuando se hace referencia a los dispositivos más pequeños y limitados. Como tales, los volúmenes de datos que cada vez son mayores, y que en muchos casos están incorrectamente manejados, hacen que los ingenieros y los investigadores dedicados a la ciberseguridad tengan dificultades en encontrar soluciones capaces de ofrecer servicios de ciberseguridad dentro de entornos y escenarios con recursos limitados. De hecho, en la última década se han visto programas con software malicioso, o malware por su abreviatura en inglés, sembrando el caos en las redes de computadores, desde las pequeñas y medianas empresas (PYME) hasta las grandes corporaciones globales. En consecuencia, los expertos de todo el mundo están preocupados por el hecho de tener millones de dispositivos conectados a Internet y protegidos de forma inadecuada. Entre esas amenazas cibernéticas, las botnets están creciendo a un ritmo alarmante, siendo responsables de peligrosos ataques contra objetivos tales como las infraestructuras críticas de cualquier país. A modo de ejemplo, en los últimos cinco años, botnets como Mirai, Conficker o Kraken han sido un objetivo principal de la comunidad de ciberseguridad.

A pesar de los esfuerzos y resultados de la comunidad científica, el malware sigue causando pérdidas y daños en las redes informáticas. Aun siendo cambiantes en su comportamiento y omnipresentes, no todos los malwares son idénticos; de hecho, difieren en alcance, en técnicas aplicadas y en efectividad. Sin embargo, existen funcionalidades que son universales y que comparten las distintas familias de malware. A modo de ejemplo, los programas maliciosos necesitan contactar con el atacante que está dirigiendo el ataque para recibir comandos (*p.ej.*, botnets), para llevar a cabo la exfiltración de datos (*p.ej.*, software espía) o para proporcionar el acceso no autorizado (*p.ej.*, troyanos de acceso remoto – del inglés Remote Access Trojan, RAT). En particular, los dispositivos infectados por una botnet necesitan comunicarse con los servidores de Comando y Control (C&C) para descargar infecciones de segunda etapa, realizar acciones maliciosas o quedar a la espera de nuevos comandos.

Durante la década pasada, la tendencia era utilizar diversas formas de comunicación (o

canales de contacto) con el grupo de criminales cibernéticos detrás del software malicioso; entre ellos, como los dispositivos infectados ya están conectados a Internet, las conexiones TCP/IP han sido objeto de un abuso generalizado. Desde la perspectiva del criminal cibernético, se ha demostrado que las conexiones directas entre los dispositivos infectados y las direcciones IP de los servidores de C&C son ineficaces. De hecho, poner esas direcciones IP en una lista negra para que sean bloqueadas es una técnica de bajo costo y fácil de aplicar, y muchas de esas listas están disponibles públicamente y actualizadas a diario (*p.ej.*, Spamhaus). Sin embargo, históricamente, los criminales cibernéticos han diseñado malwares que cuentan con varias direcciones IP, generalmente disponibles bajo un nombre de dominio totalmente calificado (del inglés Fully Qualified Domain Name, FQDN) específico y codificado dentro del propio programa malicioso, que cambian y rotan dinámicamente durante la vida útil del malware para evitar los mecanismos de protección basados en listas negras. Estos cambios rápidos de direcciones IP también son fácilmente abordables; los nombres de dominio configurados en el malware pueden ser bloqueados en unas pocas horas (una técnica conocida en inglés como sinkholing). Por lo tanto, los criminales cibernéticos han llegado a introducir módulos dedicados a contener algoritmos pseudoaleatorios como, por ejemplo, los algoritmos de generación de nombres de dominio (del inglés Domain Generation Algorithms, DGAs). Estos algoritmos incluyen fragmentos de código que, aunque diferentes en los detalles de implementación, sirven para generar nombres de dominio pseudoaleatorios conocidos como AGDs (por sus siglas en inglés de Algorithmically Generated Domain), los cuales podrían ser registrados por los criminales cibernéticos para actuar como puntos de encuentro entre los servidores C&C y los dispositivos infectados. Este paso intermedio permite que los cibercriminales puedan generar potencialmente millones de FQDNs de forma dinámica y sin tener la necesidad de registrarlos todos, ya que solo uno de ellos es suficiente para permitir la conexión entre el dispositivo y los servidores de C&C. El uso de estos DGAs se ha convertido en un enfoque recurrente y muy eficaz para eludir los equipos de ciberseguridad y los algoritmos de detección.

En la última década, la comunidad científica se ha dedicado a explorar soluciones de detección de botnets realizando herramientas basadas tanto en sistemas de reglas y heurísticas como en sistemas de aprendizaje automático. Actualmente, son tres las categorías principales de técnicas de detección de estas ciberamenazas que se suelen diseñar y desplegar en entornos tanto en producción como de investigación. Brevemente, las características de cada una de ellas se presentan a continuación:

i) En primer lugar, se encuentran los sistemas basados en reglas y heurísticas, que permiten analizar una gran cantidad de datos de manera rápida y representan la solución ideal para identificar vulnerabilidades y ataques bien conocidos. A pesar de estos beneficios, representan un sistema más bien reactivo, es decir, que necesitan que alguien defina el conjunto de reglas a añadir para detectar cada amenaza, bien conocidas de antemano.

ii) En segundo lugar, están los sistemas basados en el análisis e inspección profunda de paquetes (del inglés Deep Packet Inspection, DPI), que, a pesar de ser muy eficaces en algunos escenarios, resultan ser muy invasivos con respecto a la privacidad de los usuarios y casi completamente ineficaces en ecosistemas donde habitualmente se cifran las comunicaciones. Además, tampoco es una técnica apropiada para los grandes volúmenes de tráfico que caracterizan las redes actuales 5G, y aún menos para los previstos para las redes B5G.

iii) Finalmente, en la tercera categoría se localizan los sistemas basados en inteligencia artificial, que, a pesar de la necesidad de entrenar los modelos con grandes cantidades

de datos, resultan ser muy eficaces en la detección con una cierta confianza tanto en las amenazas conocidas como en las desconocidas. Gracias a estas propiedades, las herramientas basadas en inteligencia artificial se pueden convertir en soluciones proactivas, es decir, soluciones que, dentro de un determinado grado de confianza, son capaces de identificar amenazas nuevas o desconocidas.

La comunidad científica se ha dedicado a explorar soluciones basadas en inteligencia artificial y aprendizaje automático (del inglés Machine Learning, ML), y aunque se hayan teorizado muchos sistemas orientados al ML para la detección de los DGAs, el problema aún no ha sido resuelto del todo. Como tal, el alcance de esta tesis doctoral se centra en analizar los nombres de dominio generados por los DGAs, de cara a proporcionar un conjunto de herramientas de aprendizaje automático y metodologías respetuosas con la privacidad que ayuden a identificar estos patrones evasivos.

Antes de desarrollar más los objetivos y los avances en el estado del arte proporcionados por esta tesis doctoral, es necesario comprender la amenaza expuesta por parte de los DGAs. Así, resulta imprescindible elaborar algo más sobre su magnitud, ya que, según los informes tecnológicos más recientes el número de servidores de C&C sigue aumentando. Hay que destacar también que, desde la perspectiva de los atacantes, los DGAs son prácticos y eficientes porque causan un esfuerzo totalmente asimétrico, entre los recursos necesarios para obtener una conexión positiva entre el bot y el servidor C&C en comparación con los recursos necesarios para bloquear todos los posibles nombres de dominio maliciosos. Por un lado, el programa malicioso puede generar millones de AGDs por día, y el criminal cibernético detrás de la red solo tendrá que registrar y activar un par de ellos de forma aleatoria. Por otro lado, en contraposición, los equipos de ciberseguridad necesitan una manera inteligente de comprobar cada nombre potencial de dominio antes de decidir si la conexión es legítima o maliciosa, y finalmente aplicar la respuesta o contramedida más adecuada. Es precisamente en este escenario asimétrico donde los algoritmos de ML, como el reconocimiento de patrones, aportan una importante contribución. A este respecto, existen diversas herramientas de ML que implementan numerosos algoritmos y técnicas de aprendizaje que pueden ser desplegados para identificar los AGDs y sus eventuales "firmas" (o la ausencia de ellas).

Así, no debería de sorprender que la cantidad de tráfico generado en redes a gran escala como las redes 5G y las B5G influya directamente sobre la elección de las soluciones de detección de ciberamenazas basadas en algoritmos de ML. Por ejemplo, no es razonable tener un sistema de detección de intrusiones único y centralizado, en el pasado se ha demostrado ampliamente que las soluciones descentralizadas basadas en servicios pueden tratar de manera eficiente y efectiva volúmenes mucho mayores de datos generados por los dispositivos de los usuarios. Más en detalle, la ciberseguridad en 5G y en B5G involucra soluciones de autoorganización y autoprotección a través de servicios de seguridad desplegados en redes y entornos virtuales mediante el uso de tecnologías como SDN (del inglés, Software Defined Networks) y NFV (del inglés, Network Function Virtualization). Estos servicios de seguridad, que se definen como SECaaS (del inglés Security as a Service), se despliegan como parte de un ciclo automatizado que incluye los procesos de detección, de análisis y de mitigación de las ciberamenazas de forma colaborativa, escalable y descentralizada. Una de las características fundamentales de estos enfoques de protección es de ser independentes del entorno de despliegue. En otras palabras, en cuanto se garanticen los recursos adecuados, tanto en tema de recursos de computación como de soporte en tema de software y librerías de código, no importa si el lugar efectivo de actividad es un entorno en la nube (Cloud), en los perímetros de las redes (Edge) o en los dispositivos de los usuarios.

En el caso concreto de los SECaaS que hagan uso de ML, hay que comentar que efectivamente hay algunas fases que no son viables de ejecutar en entornos donde los recursos son limitados. Por lo tanto, es oportuno diferenciar entre las fases de entrenamiento, evaluación e inferencia. De hecho, a lo largo de esta contribución científica se han desarrollado componentes independientes del entorno de despliegue que apuntan a maximizar la diferenciación de estas fases, de manera que se puedan garantizar las condiciones más adecuadas para cada una de ellas. Así, usando por ejemplo técnicas de aprendizaje federado, cada fase puede ser optimizada en función del lugar de despliegue, ya sea en la nube, en el perímetro de la red o en los mismos dispositivos de los usuarios. De esta manera también se estarían garantizando que los datos obtenidos de los dispositivos no se transmitan más allá de lo necesario, ya que, en ciberseguridad, uno de los temas más en auge es precisamente el de limitar, en la medida de lo posible, la difusión de los datos personales a la nube.

Por lo tanto, en resumen, esta tesis doctoral se centra en el estudio de las botnets basadas en DGA a través de herramientas de ML para desplegarse como SECaaS en los perímetros de las redes 5G y B5G, con el objetivo de contribuir a la mejora del estado del arte en la identificación de aquellos elementos que permitan distinguir actividades sospechosas en entornos altamente dinámicos. En concreto, se han utilizado herramientas propias de múltiples ramas de conocimiento dentro de la inteligencia artificial y del ML, como el reconocimiento de patrones comunes en los malwares que integran los DGAs. Por aportar un ejemplo, se han estudiado, implementado y evaluado técnicas de análisis del lenguaje natural (del inglés Natural Language Processing, NLP) a fin de identificar similitudes y diferencias en la sintaxis de los AGDs. Así, esta investigación multidisciplinar abarca estudios del estado del arte, elaboración de teorías y modelos basados en aprendizaje automático, y diseño de experimentos y evaluaciones, así como la identificación y discusión de brechas de conocimiento.

Fijando como objetivo principal de la tesis doctoral el avance del conocimiento en tema de ciberseguridad y botnet basadas en DGAs, en esta tesis doctoral se han realizado dos contribuciones principales. Por un lado, esta tesis doctoral proporciona una caracterización de los aspectos de los DGAs incluyendo, entre otros, un estudio completo de contribuciones anteriores presentes en el estado del arte, fuentes de datos y enfoques basados en aprendizaje automático para la detección de botnets basadas en DGA. Por otro lado, en esta tesis doctoral se ha conseguido el objetivo aún más ambicioso de integrar y mejorar el estado del arte en términos de técnicas y literatura, proporcionando métodos, estrategias y tecnologías para permitir la detección de botnets basada en DGA a gran escala, es decir, en redes 5G y en las B5G, a través de técnicas avanzadas de ML. Específicamente, los patrones de firma que se han podido identificar en los AGDs usando técnicas de NLP y resultando en modelos de aprendizaje diseñados para ser implementados en SECaaS para que luego se puedan desplegar dinámicamente en cualquier ubicación de la red.

Metodológicamente se ha utilizado la modalidad de compendio de publicaciones para la consecución de esta tesis doctoral, siendo tres los artículos que la componen y cuyo resumen se ofrece a continuación.

- El primer artículo presenta un estudio del arte exhaustivo de los enfoques de aprendizaje automático para la detección de redes de bots basadas en DGAs. Este artículo analiza y propone algoritmos supervisados y no supervisados, sus conjuntos de características, la definición de casos de uso y experimentos, y en última instancia, el esquema de múltiples desafíos de investigación para guiar la tesis doctoral. Los hallazgos experimentales que surgen de este primer artículo sientan las bases para un estudio formal y verificable de los nombres de dominio generados por los DGAs, llamados AGDs.

- El segundo artículo aporta un análisis comparativo de las fuentes de datos para impulsar los modelos de aprendizaje automático. Este artículo informa sobre los conjuntos de datos publicados, proporcionando una comparación formal y una discusión sobre múltiples propiedades ortogonales. En el artículo también se presenta el dataset UMUDGA como la colección más completa, equilibrada y actualizada de datos relacionados con los DGAs hasta el momento de su publicación, con 50 clases de programas maliciosos para un total de más de 30 millones de FQDNs. Además, el análisis exploratorio reportado en el artículo sugiere que son factibles las soluciones de aprendizaje automático basadas en el reconocimiento de patrones y NLP para identificar con precisión variantes de programas maliciosos.
- El tercer artículo presenta una prueba de concepto donde la detección de botnets basadas en DGA se implementa como un servicio de seguridad en los perímetros más lejanos de la red. Este artículo compara y examina enfoques arquitectónicos de Edge Computing para permitir la detección escalable en redes 5G y futuras. En el artículo, la evaluación experimental demuestra que la detección de los nombres de dominio maliciosos no solo es razonable y alcanzable, sino que también es plausible esperarse a que tales capacidades de detección sean desplegadas en los perímetros de las redes, e incluso en los dispositivos de los usuarios finales.

En resumen, los artículos de investigación que componen esta tesis doctoral promueven una investigación que explora, analiza y, en última instancia, aborda las redes de bots basadas en DGA. Siguiendo este hilo conductor, cada artículo es autónomo y proporciona información crítica sobre los desafíos de la investigación desde una perspectiva diferente. En conjunto, estas contribuciones representan una descripción clara del nicho de investigación resumido en la tesis doctoral. Sin embargo, aunque concluyentes sobre los temas explorados, algunas cuestiones planteadas por esta investigación necesitan de mayor esfuerzo para su resolución. El principal de ellos es si será factible proporcionar perfiles anónimos, intercambiables y confiables para los nombres de dominio maliciosos a fin de permitir modelos de detección colaborativos y federados sin perjudicar la privacidad de los usuarios.

PhD Thesis Summary

1

Introduction and motivation

Fast performances and low latency are strict requirements for every commercial solution and research proposal, especially now that "everything" is being offered as on-demand service. Additionally, despite the millions of devices connected since the advent of commercial fifth generation (5G) platforms, a radical increase is to be expected with beyond 5G (B5G) networks [1, 2]. The problem with these devices, however, is that their cybersecurity is often overlooked. There are no strict and enforceable specifications regarding the minimum security requirements of devices, especially resource-constrained ones. As such, the ever-growing volumes of poorly handled data make researchers and engineers struggle in finding innovative-yet-reliable solutions capable of delivering cybersecurity services in resource-constrained scenarios. Indeed, the last decade has seen malwares wreaking havoc of computers networks, from small and medium enterprises (SMEs) to massive global corporations: all around the globe, the experts have been, and still are, concerned with the risks of having billions of inadequately protected devices connected to the internet [3].

Besides being diffuse and pervasive, malwares differ in scope, applied techniques, and effectiveness. However, there exists a single universal functionality that is shared among them, *i.e.*, any malware needs to reach out to the owner for commands (*e.g.*, botnets), exfiltrate data (*e.g.*, spyware), or provide unauthorised access (*e.g.*, remote access trojans (RATs)) among others. The past decade trend was to use various communication channels to contact the cybercriminal group behind the malware; among them, HTTP(s)-based connections are the most common technique. Although effective in some scenarios, deep packet inspections (DPIs) and other content-based techniques are impractical in an ecosystem where connections are encrypted more often than not. Essentially, they are not suitable for the large volumes that characterise today's 5G networks [4], let alone the envisioned numbers that will characterise B5G networks [1].

From the cybercriminal perspective, direct IP connections between the infected devices and the Command & Control (C&C) servers have been proved ineffective. Indeed, blacklisting such addresses is a low-cost, practical, and well-known technique; furthermore, daily-updated offenders' lists are publicly available (*e.g.*, Spamhaus [5]). Cybercriminals have historically designed malwares that feature multiple IP addresses (generally available under a specific and hardcoded fully qualified domain name (FQDN)) that dynamically change and rotate during the malware's expected lifespan to bypass this protection mechanism. However, these fast-flux IP addresses are also somewhat easy to tackle, as the rendezvous domain name can be blocked in just a few hours (a technique known as sinkholing). Hence, cybercriminals came up with the dynamic generation of FQDNs via pseudo-random generation modules within the malware code. Such modules contain a

domain generation algorithm (DGA), a fragment of code that, although different in the implementation details, serves to generate pseudo-random domain names that might be registered by the cybercriminals to act as *rendezvous-points* between C&C servers and infected devices. This intermediate step permits the cybercriminals to generate millions of FQDNs dynamically without the necessity to register all of them; in fact, one available domain name is just enough to permit the connection between the infected device and the C&C servers [6].

To grasp the threat, it is imperative to understand its magnitude before all else; in fact, according to the most recent tech reports [5, 7], the number of botnets C&C is still increasing. From the cybercriminal point of view, DGAs are practical and efficient because of the asymmetrical effort required in contrasting them. On the one hand, a single malware variant can generate millions of algorithmically generated domains (AGDs) per day, having only a few of them registered and active; however, on the other hand, the security teams need to check each FQDN and decide the most appropriate response. In this precise scenario, artificial intelligence (AI), and precisely machine learning (ML) algorithms such as patterns recognition, show their potential. The ML toolbox offers a set of tools and techniques that can be deployed to identify the DGAs variants' signature (or their lack of); section I will further discuss the subject. Indeed, in 2019 the number of AGDs registered plummed [5], confirming the excellent research results published in the previous five years [6, 8, 9, 10, 11, 12, 13]. Nonetheless, domain registrars' defensive policies fall short when vetting the registration of AGDs, hence permitting the abuse to continue [7].

This PhD thesis focuses on studying these generation algorithms with machine learning tools to identify elements that can distinguish suspicious activities in highly dynamic 5G/B5G environments. Indeed, ML has been studied and successfully deployed to recognise common patterns in generated domains, often leveraging syntactic analysis from Natural Language Processing (NLP). As such, the first chapter of this thesis (Scalable detection of botnets based on DGA (Article 1–`SoCo`)) focuses on surveying the literature aiming to establish a trend in algorithms and, in general, machine learning applications. The second chapter (UMUDGA: A dataset for profiling DGA-based botnet (Article 2–`CoSe`)), however, focuses on the data sources used to train and validate these models; after a thorough review of the available references, it unveils the University of Murcia Domain Generation Algorithm Dataset (`UMUDGA`), a collection of 30+ million domain names and 50 malware variants. Finally, the third chapter (Early DGA-based botnet identification (Article 3–`Clus`)) further advances these subjects by focusing on architectures that can maximise massive detection performance while minimising privacy leakage.

Indeed, we deem necessary to differentiate two research questions, namely identifying malicious and legitimate domain names using ML techniques, and the capability to do so at scale. The aspects related to the former research question (*i.e.*, the identification of AGDs) are discussed and analysed in section I, while, the latter's challenges (*e.g.*, the identification of AGDs at scale) require introducing the edge intelligence (EI) paradigm [14, 15] and how DGAs detection might benefit from its usage. The EI's concept has been widely discussed under different names (*e.g.*, mobile computing and fog computing, among others, as reported in the third chapter of this thesis, Article 3–`Clus`). The key is that its fundamental principle still applies to the subject at hand, *i.e.*, the detection process as a collection of decentralised micro-services that benefit from a shared knowledgebase [15, 16]. At the core of the EI paradigm, to put it differently, there is exploring the synergies between the cloud services, the decentralised and often automated edges, and the user equipments (UEs). In the application boundaries defined by the DGA-detection, this collaboration outlines the ML components' separation into virtual services, available

on-demand, that shares data or trained models. As such, legitimate and suspicious FQDNs might be gathered and identified locally and then shared (anonymously) with the network of detection modules or a centralised authority. Shared data can be used to improve a pre-trained model's detection performances, either centralised in the cloud or spread over several independent on-edge detectors.

In the third chapter (Article 3–`Clus`), various architectural approaches to achieve such a synergy have been studied and discussed. In essence, a detection framework should consider to discuss and eventually balance the desired detection performances, users' privacy, and agility required to face new and unknown threats. As a point of fact, the learning models' quality relies on the quality of the collected data, which ultimately needs to be detached and anonymised regarding the user base to prevent privacy-related issues, without losing the capability to represent the deployment environment.

Ideally, a transparent view of the data will result in models with improved detection rates capable of identifying known anomalies and new threats. Despite the anonymisation introduced in such a collaborative environment, it remains unclear if it is possible to profile and uniquely identify the users, thus increasing the risk of personal data exposure. On paper, provided that the data security is guaranteed, collaborative learning models permits to achieve the scalability required by the volumes involved with 5G networks (and beyond), without losing the agility needed to face newly identified threats. These concepts are further discussed and analysed in section II.

I Domain name identification as a machine learning task

Differentiate malicious AGDs from legitimate FQDNs can be seen as a pure ML task, independent from the actual use case or application environment. In such a scenario, key performances are the classification ones (*e.g.*, precision and recall, among others) rather than training and testing time or resources requirements. Under this prism, deep learning (DL) techniques provide good results without requiring too much work on the data preprocessing; however, their nature of "black box" algorithms make them of difficult interpretation, especially when it comes to making sense of the outcome results. As the literature suggests (*cf.*, Article 1–`SoCo`, Section 2) and experiments demonstrate (*cf.*, Article 1–`SoCo`, Section 3.3, Article 2–`CoSe`, Section 4, Article 3–`Clus`, Section 3), the Neural Networks (NNs)' sophistication is not required to tackle the DGA-related challenges. In other words, the collected AGDs call for a straightforward feature engineering process, rather than the automated, self-learning approach where DL stands out.

Furthermore, in a real-world application scenario where data sharing has to be examined (and its ramification discussed), collecting vast amounts of data required for training DL models could be difficult. On the contrary, classical ML approaches suffer less from the lack of sufficiently large, precisely labelled datasets. For another thing, as previously stated, one could argue that DL solutions could potentially work without features, hence simplifying the preprocessing steps required to process the data; however, the features identification process will still be carried out by the first layers of the network, that, in turn, require training. The DL training phase might also carry challenges related to required training resources and the amount of data required to feed the model. In the end, the data itself has to be encoded and eventually scaled or normalised before usage, thus impairing the benefits of avoiding a predefined feature extraction process. Thus, the application of classical ML algorithms, rather than DL ones, has been deemed more suitable to the task at hand.

Besides, one could argue that the DGA-related challenges have been sufficiently dis-

cussed in the literature, and thus solved. However, as demonstrated in this thesis, this is not the case. Essentially, each chapter of this thesis unveil critical shortcomings of the previously published frameworks and results, ultimately pointing out at yet-unsolved research challenges. Indeed, three main themes should be evaluated, namely *i)* the feature engineering process, *ii)* the data sources, and *iii)* the applied learning model. These aspects will be discussed in the following paragraphs.

Feature analysis — One of the essential aspects drawn from this thesis' feature analysis has been identifying two families of features, namely the context-aware and the context-free ones. In other words, the gathered features have been divided into two families, depending on whether they include (or not) users' personal and behavioural data. For example, packet-inspection -related features (such as time-based ones) are included in the context-aware family; on the contrary, NLP features (such as the ratio between characters) are included in the context-free family. The first chapter of this thesis (Article 1–`SoCo`) is dedicated to the literature review of these features' aspects; however, the feature formalisation and implementation has been studied in the second chapter of the thesis (*cf.*, Article 2–`CoSe`, Section 3).

Across this thesis, though, it has been proved that most of the gathered features are not adequate in describing the data. To be precise, in the first chapter (*cf.*, Article 1–`SoCo`, Section 3), several feature selection and feature extraction algorithms have been applied, leading to demonstrate that, in general, only a handful of features are needed to classify the AGDs correctly. Furthermore, in the second chapter (*cf.*, Article 2–`CoSe`, Section 4), it has been shown that, albeit using the full set of features, AGDs generated by some variants are indistinguishable. Likewise, the third chapter (*cf.*, Article 3–`Clus`, Table 2 and Article 3–`Clus`, Table 3), pointed out that limiting the feature set to the top 10 most informative ones significantly improves the resource consumption without unreasonably hindering the classification performances.

Data sources — For another aspect, there is a general lack of publicly available and extensive datasets regarding AGDs (and FQDNs in general). As firstly identified in the first chapter of this thesis (Article 1–`SoCo`), a quantitative comparison between the proposed frameworks is unfeasible, mainly due to the lack of shared data. The second chapter of this thesis (*cf.*, Article 2–`CoSe`, Table 1) focuses on surveying published and well-recognised data sources related to network traffic to address this shortcoming. In the article, nine orthogonal metrics have been designed, discussed and studied to present a formal comparison of the data sources available in the literature. The analysis led to the publication of a new data source that satisfies all the identified properties. Specifically, the second chapter presents both the data collection framework and the methodology followed to create the dataset (*cf.*, Article 2–`CoSe`, Section 3).

Learning models — Finally, several attempts have been made on the subject of the ML models for AGDs identification. As presented in the first chapter of this thesis, the literary review manifests the community's interest in exploring several potential algorithms (*cf.*, Article 1–`SoCo`, Section 2). In the article, several ML frameworks have been identified from the literature, leveraging supervised, unsupervised and mixed approaches. However, across the scale surfaced a general lack of reproducibility in terms of data, features and algorithms configurations.

For instance, as reported in the first chapter of this thesis (Article 1–`SoCo`), most literature claim to achieve good to excellent classification results, without providing information

regarding either the data, the preprocessing, or the models' hyperparameters. As such, the first chapter pivots on two main contributions: providing a complete list of tested approaches (in the form of a comparative analysis of algorithms and claimed results) and an exhaustive list of studied features.

On the contrary, the second chapter analyses the dataset with five among the most common ML algorithms. The resulting exploratory analysis provides useful insights into the dataset composition, classes, and properties (*cf.*, Article 2–`CoSe`, Section 4).

Finally, the third chapter (Article 3–`Clus`) explores the capabilities of lightweight, tree-based classifiers at scale. To be precise, a collaborative framework is theorised, designed, and implemented leveraging the first two chapters' results (Article 1–`SoCo` and Article 2–`CoSe`). In such a scenario, the main focus is on the users' privacy; indeed, a shared-intelligence system can be designed based on the federated learning theory to provide anonymous and shareable knowledge without having to share users' data.

II Cybersecurity as a service

Besides the challenging aspects of the ML tasks, the application use cases offer another perspective. Indeed, when deploying intrusion detection systems (IDSs), it is imperative to find the balance between detection performances and resources requirements. In other words, it is often convenient to accept a reduction in the detection rate in return for higher traffic volumes. In such a context, metrics like the classification yield or the resources consumption assume a higher relevance than in the pure ML tasks, and further optimisations are required to determine the equilibrium.

Indeed, the detection probes' locations influence, as expected, the amount of traffic that can be successfully inspected. For example, it is unreasonable to have a single and centralised detection module, especially when considering the volumes and the mobility requirements of 5G networks and beyond. On the contrary, multiple and decentralised IDSs are often deployed to scale the protection modules to cover a higher volume of connected devices. To be precise, the scalability-related challenges have been widely explored in the past, each time regarding the application scenario's specific requirements–mobile and not. However, with the explosive increase of connected devices related to the 5G and B5G networks, the paradigm of the security-as-a-service (SECaaS) becomes critical.

Traditionally, the cybersecurity approach to 5G involved the network components' self-organisation using software-defined network (SDN) and network functions virtualization (NFV) technologies. The virtualised services are in a cycle where the processes of detection, analysis, and mitigation of security threats work simultaneously and in a coordinated fashion. In such a scenario, this thesis aligns with the detection and analysis services by exploring, theorising, and discussing architectural designs to offer DGA-botnet detection as a dynamic module compatible with modern networks' strict requirements. Therefore, the concept of EI has been investigated and combined with the federated learning theory, ultimately proving that the detection process is not only feasible on the networks' farthest edges, but also on UEs. Indeed, the third and final chapter of this thesis (Article 3–`Clus`) focuses on these aspects of detecting and identifying DGA-based malwares leveraging the EI theory.

In the third article (*cf.*, Article 3–`Clus`, Section 2.2), several EI-compatible architectures are identified and studied, aiming to establish a knowledgebase useful to deploy DGA-detection modules by decoupling the training and testing phases from the model inference process. By virtualising these processes as services, mixed cloud-edge-device scenarios become available via technologies such as the SDN and NFV. Indeed, throughout

this thesis, the main design principle has been to provide modular services that can be plugged in as independent components in a complex framework such as the ones proposed in 5G/B5G researches. By guaranteeing the separation between the learning process elements and phases, the compatibility with the SECaaS paradigm is achieved, and, as a consequence, whether the detection modules are designed and realised internally or outsourced becomes a secondary question. Indeed, as pointed out in the first chapter of this thesis (Article 1–`SoCo`), the Context-Free features used throughout the research can be extracted from the collected FQDNs independently from the actual network traffic, inspection techniques or technology implementation. Similarly, the second chapter's principal contribution (Article 2–`CoSe`) (*i.e.*, the `UMUDGA` dataset) might benefit any SECaaS provider by providing *i)* the high-quality labelled data to train ML models, and, *ii)* the testing data to be injected to evaluate the framework's performances.

As a consequence of the SECaaS paradigm compatibility, properties such as the managed execution and isolation can be guaranteed at any moment. The former ensures that the services can be deployed independently and where they are needed the most. On this subject, the third chapter of the thesis (*cf.*, Article 3–`Clus`, Figure 2) explores the different configurations from cloud to edge deployment, and eventually inferred on-device compatibility. Similarly, the latter property can be enforced by carefully deploying the services through containerisation, preventing unauthorised access to the model and the data itself. In this context, the potential capabilities hinted by developing federated learning solutions might permit to decouple the private data and models from the shared re-trained models. This subject has been defined and discussed, along with the privacy-related aspects and challenges, in the third chapter (Article 3–`Clus`).

III Objectives

Identifying AGDs at scale in complex-environment, such as the ones offered by 5G/B5G scenarios, pointed out several challenges and multiple criticalities that had to be addressed.

As such, the first objective, defined as follows, aims to identify where the state-of-the-art is in terms of DGA-based botnet detection, with special attention to ML approaches, data sources, and published frameworks.

Objective 1: State-of-the-art (`O1-SoTA`). Outline and study the aspects of DGA-based botnet detection in 5G and B5G scenarios.

As the first objective is broad in both scope and potential approaches, four sub-objectives have been outlined; to be precise, the first three objectives aim to identify and explore the state-of-the-art under different prisms, while the last one aims to draw the needed conclusions necessary to identify the research path.

Objective 1.1. Study and present a critical revision of researches on DGA-based botnet detection in 5G/B5G scenarios previously published in high-quality journals and conferences.

Objective 1.2. Study and present a critical revision of publicly available data sources to power ML detection frameworks.

Objective 1.3. Collection and analysis of published ML solutions to identify samples of AGDs.

Objective 1.4. Identify and present research gaps, challenges, and potential future lines.

Following the identification and discussion of the characteristics of the problem at hand, the second stage of this PhD thesis focuses on achieving a more hands-on objective. Indeed, starting from state-of-the-art ML solutions and approaches, combining them with the obtained data sources, and having the 5G/B5G as the primary use case, a novel, scalable, and service-oriented framework has been designed to achieve the next objective:

Objective 2: Identification framework (`O2-FRMW`). Theorise, design and implement a proof of concept ML-based and SECaaS-compatible identification framework for DGA-based botnet detection in 5G and B5G scenarios.

Similarly, four steps and milestones have been identified to achieve this second objective:

Objective 2.1. Obtain and maintain a curated data source to enable reproducible ML experiments.

Objective 2.2. Study and present architectural approaches to the detection of DGA-based botnets in modern networks.

Objective 2.3. Design, implement, and evaluate ML approaches to DGA-based botnet identification.

Objective 2.4. Design, implement, and evaluate a SECaaS container to enable ML detection in a collaborative environment.

Besides the two primary objectives just identified, a series of methodological principles to maintain the research on a scientific and reproducible track throughout the PhD thesis has been identified and formalised as follows.

Principle 1*:* **Reproducibility of the research (`P1-REPR`)**. Each research contribution shall be studied, formalised with specific attention to those details that enable to replicate and validate the findings and experiments.

Principle 2*:* **Open science (`P2-OPEN`)**. Each research contribution shall be publicly released, including knowledge, data, and source code.

Principle 3*:* **Keep-it-simple (`P3-KIS`)**. Each research contribution shall be unravelled and reduced to essential components, to facilitate reproducibility, evaluation, and usage of the obtained result.

Principle 4*:* **Evaluation and validation of the results (`P4-EVAL`)**. Each research contribution shall be published in high-ranking peer-reviewed venues.

2

Methodology

This PhD thesis has been conducted following a scientific approach based on researching the state-of-the-art, from which key points, challenges, and theorised solutions have been proposed. As a result of the first initial literary review, it has been made clear that several issues made it impracticable to reproduce and validate numerous published results. Therefore, any subsequent effort has focused on enabling research outcomes' quantitative and qualitative rigorous analyses and comparisons. Throughout the research, the methodology aimed at adhering to the three principles defined in the previous section, namely `P1-REPR`, `P2-OPEN`[1], and `P3-KIS`. By publishing these results, this thesis also heeds to the fourth principle, *i.e.*, `P4-EVAL`.

It is well known that it is imperative to match originals conditions to replicate any experiment or result. When depicted in ML solutions, the `P1-REPR` principle implies to deploy models with the same configurations using the same data sources (both raw data and preprocessing). Under such a prism, and aligned with the `P1-REPR` and `P2-OPEN` principles that guide this thesis, the first two chapters (Article 1–`SoCo` and Article 2–`CoSe`) address precisely the data sources, their elaboration and their analysis using ML models. Hence, to achieve the `O1-SoTA` objective, the first chapter identifies and collects several research articles published in the previous five years on the subject of DGA-based botnet detection.

The analysis of the state-of-the-art, as requested by the `O1-SoTA` objective, led to the definition and formalisation of the features sets proposed in the literature, divided into two general families (*cf.*, Article 1–`SoCo`, Section 2), *i.e.*, those that rely on users' data (Context-Aware features) and those that rely only on the domain name itself (Context-Free features), being the latter the most common one deployed. These Context-Free features have been studied, reimplemented, and evaluated in the first chapter of this thesis (Article 1–`SoCo`, reaching `O1.1` sub-objective), and eventually formalised and published within the `UMUDGA` dataset (Article 2–`CoSe`, reaching sub-objectives `O1.3` and `O2.1`). The conclusions that have been drawn from the knowledge acquired on this subject hinted that authors have previously theorised new features without pondering whether they provide enough information to justify the computation resources needed to calculate them (sub-objective `O1.4`). For example, as reported in the first chapter (Article 1–`SoCo`), considering metrics related to nGrams distributions with $n \leq 2$ requires to add a non-trivial amount of resources to the feature extraction process as there are $|S|^n$ valid symbols per distribution, where S is the set of valid ASCII symbols for domain names and n is the size of the analysed nGram.

[1] Although the research has not been published under the Open Access modality, each article has been legally released without restrictions as pre-print copy in the appropriate repositories.

Furthermore, as the sequence of characters taken into account increases, the probability of having such combination represented in a domain name decreases, leading to mostly zeroed distributions.

Once again, the keep-it-simple principle (P3-KIS) is reflected in the methodology and results. Indeed, all three chapters (Article 1–SoCo, Article 2–CoSe, and Article 3–Clus) demonstrated that privacy-aware, syntactic analysis of the domain names is enough to achieve outstanding classification performances, without the need to explore elaborated metrics that require profiling the users' behaviour. In the first chapter of this thesis, an exploratory analysis of the data using different feature selection and extraction techniques is carried out to prove the results (sub-objectives O1.3 and O2.3). In the experiments, six among the most famous (and used in the literature) classifiers have been deployed to identify the DGAs variants, suggesting that most of the identified features are indeed not providing a sufficient amount of information. In each chapter (Article 1–SoCo, Article 2–CoSe, and Article 3–Clus) the data, algorithms, and evaluation results have been thoroughly described and formalised (P1-REPR principle). However, it is yet to be discussed if solutions using Context-Aware features can outperform the already excellent results obtained in the experiments (O1.4 sub-objective).

The literary review also pinpointed that the already published researches were carried out mostly without releasing the data sources (sub-objective O1.3 and principle P2-OPEN). Hence, this thesis's second chapter focuses on the formal analysis between the published resources and the newly presented UMUDGA dataset (sub-objectives O1.3 and O2.1, P2-OPEN principle). Furthermore, a comparison framework has been designed to achieve sub-objective O1.2; in the framework, nine orthogonal metrics summarise the different properties, advantages and disadvantages of each data source, ultimately highlighting their lack of completeness. Among the metrics presented in the second chapter (Article 2–CoSe), it is possible to pinpoint the sources' verifiability, the data extensibility, and the ML readiness. Among others, these metrics suggested once again the critical importance of reproducible (P1-REPR principle) and usable open data (P2-OPEN principle). Indeed, the data collection process has been carried out by collecting the numerous malware variants from publicly available sources (such as previous researches, tech blogs, or vendor-specific bulletins, to cite a few) and executed it with predefined seeds (to control the output deterministically) to collect the resulting AGDs (sub-objective O2.1).

Furthermore, to complete the achievement of the O1-SoTA objective, a literary review of different architectural designs for DGA-based botnet detection at scale has been conducted. The last chapter (*cf.*, Article 3–Clus, Section 2.2) provides a technical overview of the different SECaaS architectures for ML-based detections on the network edges (sub-objectives O1.1 and O1.3). The article explores the different properties of the detection probes' location, providing the critical points to foster the discussion regarding the trade-off between classification capabilities, resource constraints, and limitations in user data usage.

Similarly, the O2-FRMW objective requested components and modules can be identified in each of the chapters composing this thesis. Indeed, following a bottom-up approach, the first and foremost element can be represented by the malwares' DGAs source code. As described in the second chapter (*cf.*, Article 2–CoSe, Section 3), and to achieve sub-objective O2.1, 50 malware variants have been collected; hence, their DGAs have been reimplemented and executed to generate at least 10.000 AGDs per variant (most DGAs, however, have been used to collect one million samples). As a result, the collected data, renamed as the "UMUDGA dataset", has been processed and publicly released after a peer-review process (Article 2–CoSe) and have been used to feed any experiment publicly released in this thesis

(sub-objective O2.1, P1-REPR and P2-OPEN principles). Besides, with the formalisation of the studied features (Article 2–CoSe), each methodological principle can be identified: the designed methodologies and deployed procedures have been formalised in all research contributions and have been evaluated by the scientific community (P1-REPR and P4-EVAL principles); the source code has been released for each step of the research, including proof-of-concepts services and algorithms (P1-REPR and P2-OPEN principles); multiple approaches and solutions have been studied and ultimately formalised in the publications, proving that straightforward approaches can achieve excellent results without the need for obscure or uninterpretable constructs (P1-REPR and P3-KIS principles); each contribution has been published in competitive, high-quality, and peer-reviewed journals (P4-EVAL principle), namely Soft Computing (Article 1–SoCo), Computers & Security (Article 2–CoSe), and Cluster Computing (Article 3–Clus).

Indeed, the experimental detection modules have been studied and developed since the first (Article 1–SoCo) and second chapters (Article 2–CoSe), in which six among the most used classifiers have been described, implemented, trained and tested (O2.3). In these publications, the detection modules are examined as machine learning solutions, thus focusing on the classification performances in different scenarios and with different data sources. However, the third chapter (*cf.*, Article 3–Clus, Section 3) investigates the detection module as a service (O2.4), highlighting and discussing properties such as the requirements and performances, service deployment location, and eventually, how the detection probes relate to the users' privacy (O2.2). Finally, throughout these contributions, it is possible to identify the keep-it-simple principle (P3-KIS); for example, although deep and convoluted DL architectures have been proven effective, we have demonstrated that many intelligible and straightforward solutions can be equally valid.

3

Conclusions and future work

Formally, this PhD dissertation aims at providing a study on domain generation algorithms (DGAs), and specifically on the techniques that can be used to identify them in the wild. However, this research's unwritten objective is to untangle the amount of machine learning (ML)-based contributions (that claims to solve the algorithmically generated domains (AGDs) identification problem) to describe a straightforward, working, and scalable approach that does not jeopardise users' privacy. As such, guided by keep-it-simple (`P3-KIS`)'s principle, this PhD thesis surveyed the literature regarding algorithms, data sources, and frameworks to DGA-based botnet detection in fifth generation (5G) networks and beyond. Indeed, security-as-a-service (SECaaS) and edge intelligence (EI) have been studied and applied to provide the required dynamicity characteristic of these modern environments.

Among the leading contributions achieved, three specifically outshine the others:

i) firstly, the formalisation of the designed features used to build ML solutions oriented to the detection of DGA-based botnet, with particular attention to those that do not require users' profiling (*cf.* Article 1: Scalable detection of botnets based on DGA);

ii) secondly, the collection and public release of a complete and up-to-date dataset for DGA-based malwares, including 50 DGAs and over 30 million AGDs (*cf.* Article 2: UMUDGA: A dataset for profiling DGA-based botnet); and,

iii) thirdly, the proof-of-concept AGDs detection module's design and implementation devised to be deployed as on-edge SECaaS (*cf.* Article 3: Early DGA-based botnet identification).

As per the evaluation and validation of the results (`P4-EVAL`) principle, these primary contributions have been published in top-tier journals, namely two Q2 (*Soft Computing* for Article 1 and *Computers & Security* for Article 2) and a Q1 (*Cluster Computing* for Article 3). Nowadays, research in computer science, particularly in cybersecurity and machine learning, still suffers from the lack of reproducibility. Indeed, many results and solutions claim to achieve exceptional results, and while that might be the case, the replicability and validation processes are often overlooked. As such, each contribution presented in this PhD thesis adhered to the principles of reproducibility of the research (`P1-REPR`) and open science (`P2-OPEN`). In doing so, the achieved results might be evaluated, reimplemented, and eventually improved by the scientific community and future researchers.

Together, they deliver the PhD dissertation's established objectives; however, some challenges are yet to be solved.

For one thing, this research focuses on analysing those features that do not require users profiling (context-free); however, their context-aware counterpart has been suggested as equally valuable in the literature as, intuitively, each malware variants behaves differently. As such, future works might reveal that a combination of these two families upholds the key to identify the most advanced malwares. Besides the potential advantages offered by context-aware features, it is imperative to consider that they require more invasive techniques than their context-free analogue. As such, future research should focus on how to provide AGDs detection services without harming users' privacy.

Furthermore, following the privacy subject, a specific issue strikes out: any given model needs to be trained with real-world data to learn its environment's characteristics. In such a context, sharing knowledge becomes a critical feature of collaborative detection frameworks; in particular, we hinted at the federated learning capabilities, without delving too much into it. Noteworthy future researches might explore this and other cooperation paradigms to unveil innovative solutions for identifying AGDs without having to share users data.

Last but not least, a remark is needed. Frameworks and proposals are designed, analysed, tested, and discussed in specific minimal use cases and scenarios. Despite this condition, research often overlooks the validation process and comparison with other previously published results. As such, the research area is in great need of a proper formal suite of validation benchmarks, to which new solutions should adapt.

Bibliography

[1] Q. V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W. J. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5G and beyond: fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116 974–117 017, 2020. DOI: 10.1109/access.2020.3001277

[2] C. Benzaid and T. Taleb, "AI for beyond 5G networks: a cyber-security defense or offense enabler?" *IEEE Network*, vol. 34, no. 6, pp. 140–147, Nov. 2020. DOI: 10.1109/mnet.011.2000088

[3] European Union Agency for Network and Information Security, "Malware threat landscape 2020," European Union Agency for Network and Information Security, Tech. Rep., 2020. [Online]. Available: https://www.enisa.europa.eu/topics/threat-risk-management/threats-and-trends/etl-review-folder/etl-2020-malware

[4] D. Papamartzivanos, F. Gómez Mármol, and G. Kambourakis, "Introducing deep learning self-adaptive misuse network intrusion detection systems," *IEEE Access*, vol. 7, pp. 13 546–13 560, 2019. DOI: 10.1109/access.2019.2893871

[5] Spamhaus Malware Labs, "Botnet threat report 2019," The Spamhaus Project SLU., Tech. Rep., 2019. [Online]. Available: https://www.spamhaustech.com/botnet-threat-report-2019/

[6] A. Khormali, J. Park, H. Alasmary, A. Anwar, M. Saad, and D. Mohaisen, "Domain name system security and privacy: a contemporary survey," *Computer Networks*, vol. 185, p. 107699, 2021. DOI: 10.1016/j.comnet.2020.107699

[7] Spamhaus Malware Labs, "Botnet threat update: Q1-2020," The Spamhaus Project SLU., Tech. Rep., 2020. [Online]. Available: https://www.spamhaus.org/news/article/798/spamhaus-botnet-threat-update-q1-2020

[8] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla, "A comprehensive measurement study of domain generating malware," in *25th USENIX Security Symposium*, Austin, TX, Aug. 2016, pp. 263–278. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/plohmann

[9] X. Luo, L. Wang, Z. Xu, J. Yang, M. Sun, and J. Wang, "DGASensor: fast detection for DGA-based malwares," in *5th International Conference on Communications and Broadband Networking*, Bali, Indonesia, Feb. 2017, pp. 47–53. DOI: 10.1145/3057109.3057112

[10] H. Mac, D. Tran, V. Tong, L. G. Nguyen, and H. A. Tran, "DGA botnet detection using supervised learning methods," in *8th International Symposium on Information and Communication Technology*, Nha Trang City, Viet Nam, Dec. 2017, pp. 211–218. DOI: 10.1145/3155133.3155166

[11] D. Tran, H. Mac, V. Tong, H. H. A. Tran, and L. G. L. Nguyen, "A LSTM based framework for handling multiclass imbalance in DGA botnet detection," *Neurocomputing*, vol. 275, pp. 2401–2413, 2018. DOI: 10.1016/j.neucom.2017.11.018

[12] R. R. Curtin, A. B. Gardner, S. Grzonkowski, A. Kleymenov, and A. Mosquera, "Detecting DGA domains with recurrent neural networks and side information," in *14th International Conference on Availability, Reliability and Security*, Canterbury CA, United Kingdom, Oct. 2019, pp. 1–10. DOI: 10.1145/3339252.3339258

[13] C. Catania, S. García, and P. Torres, "Deep convolutional neural networks for DGA detection," in *Computer Science – CACIC 2018*, Tandil, Argentina, Oct. 2019, pp. 327–340. DOI: 10.1007/978-3-030-20787-8_23

[14] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: the confluence of edge computing and artificial intelligence," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7457–7469, 2020. DOI: 10.1109/jiot.2020.2984887

[15] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: a comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020. DOI: 10.1109/comst.2020.2970550

[16] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: a comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020. DOI: 10.1109/comst.2020.2986024

Publications composing

the PhD Thesis

1

Scalable detection of botnets based on DGA

Title:	Scalable detection of botnets based on DGA: *efficient feature discovery process in machine learning techniques*
Authors:	Mattia Zago, Manuel Gil Pérez, Gregorio Martínez Pérez
Journal:	Soft Computing
J.I.F.:	3.050 Q2 (2019)
Publisher:	Springer
Volume:	24
Pages:	5517–5537
Year:	2020
Month:	January
DOI:	10.1007/s00500-018-03703-8
Status:	Published

Abstract

Botnets are evolving and their covert modus operandi, based on cloud technologies such as the virtualization and the dynamic fast-flux addressing, has been proved challenging for classic Intrusion Detection Systems and even the so-called Next-Generation Firewalls. Moreover, Dynamic Addressing has been spotted in the wild in combination with pseudo-random Domain names Generation Algorithm (DGA), ultimately leading to an extremely accurate and effective disguise technique. Although these concealing methods have been exposed and analysed to great extent in the past decade, the literature lacks some important conclusions and common ground knowledge, especially when it comes to Machine Learning solutions. This research horizontally navigates the state-of-the-art aiming to polish the feature discovery process, which is the single most time-consuming part of any Machine Learning approach. Results show that only a minor fraction of the defined features are indeed practical and informative, especially when considering zero day (0-day) botnet identification. The contributions described in this article will ease the detection process, ultimately enabling improved and more scalable solutions for DGA-based botnets detection.

Keywords

Botnet · Domain Generation Algorithm · DGA · Machine Learning · Natural Language Processing

Soft Computing (2020) 24:5517–5537
https://doi.org/10.1007/s00500-018-03703-8

FOCUS

Scalable detection of botnets based on DGA

Efficient feature discovery process in machine learning techniques

Mattia Zago[1] · Manuel Gil Pérez[1] · Gregorio Martínez Pérez[1]

Published online: 18 January 2019

Abstract
Botnets are evolving, and their covert modus operandi, based on cloud technologies such as the virtualisation and the dynamic fast-flux addressing, has been proved challenging for classic intrusion detection systems and even the so-called next-generation firewalls. Moreover, dynamic addressing has been spotted in the wild in combination with pseudo-random domain names generation algorithm (DGA), ultimately leading to an extremely accurate and effective disguise technique. Although these concealing methods have been exposed and analysed to great extent in the past decade, the literature lacks some important conclusions and common-ground knowledge, especially when it comes to Machine Learning (ML) solutions. This research horizontally navigates the state of the art aiming to polish the feature discovery process, which is the single most time-consuming part of any ML approach. Results show that only a minor fraction of the defined features are indeed practical and informative, especially when considering 0-day botnet identification. The contributions described in this article will ease the detection process, ultimately enabling improved and more scalable solutions for DGA-based botnets detection.

Keywords Botnet · Domain generation algorithm · DGA · Machine Learning · Natural language processing

1 Introduction and motivation

Computer networks enable sharing resources, immediate communications and distributed computation, which have been strengthened in recent years by the Cloud Computing paradigm where "everything" is being offered as an on-demand service (Mell and Grance 2011). Unfortunately, such network functionalities are misused by malicious entities aiming to compromise as many systems as possible in order to have them available for later use. Such compromised hosts are generically defined as *zombies* (or bots), being part of a network controlled by one or multiple command & control (C&C) servers. In order to conceal the infection and hide the botnet purposes, malwares are employing a variety of techniques such as code obfuscation and encryption (Vormayr et al. 2017; Gupta et al. 2016). Nevertheless, bots need to reach the C&C servers so as to receive commands and pull-out recorded data. There is a potential weak point from attackers perspective: once the C&C is taken out, or seized by the authorities, the botmaster–botnet owner will lose the control over the botnet (Leelasankar et al. 2018; Lerner 2014).

As a consequence, malwares are actively employing advanced evasion techniques to conceal the communications with C&C servers. A domain generation algorithm (DGA) represents a practical solution from the attacker's point of view. With DGAs, we indicate a family of algorithms that given a seed, often shipped with the malware as a pre-shared secret, generate strings of domain names that can be queried and resolved for locating the active C&C server. In addition, these evasion characteristics are currently being enhanced by using the cloud computing environment (Sharieh and Albdour 2017; Stergiou et al. 2018), in which both bots and C&C

Communicated by B. B. Gupta.

Gregorio Martínez Pérez
gregorio@um.es

Mattia Zago
mattia.zago@um.es

Manuel Gil Pérez
mgilperez@um.es

[1] Department of Communications and Information Engineering, Faculty of Computer Science, University of Murcia, Campus de Espinardo s/n, 30100 Murcia, Spain

servers can be provisioned dynamically and being moved from one location to another or even between providers within the cloud (Bugiel et al. 2011; Hussain et al. 2017). This fact makes botnets more difficult to trace and detect in real time (Zhang et al. 2014). Moreover, cyber-criminals are continuously updating their malwares' DGA in order to evade regular patterns and signatures created by security vendors. These updated versions are considered as 0-day threats. To be more precise, a 0-day is a threat that was previously unknown, which can be either in form of a new variant of a known malware or an entirely new malware family.

Most DGAs algorithms are both time-dependent and deterministic, i.e. their generation parameters are retrievable and reusable to compute all the possible outcomes. Hypothetically, it is possible to reverse engineer each malware variant to obtain the generation algorithm and the seed, thus extracting the subset of valid domain names (DNs) for a given date and time. This approach is not viable when considering the number of malware families and variants (0-days) discovered every day. Moreover, even excluding exceedingly large algorithmically generated domains (AGDs) produced by malwares such as Conficker or Virut, which can generate an average of 50,000 domains per day (Plohmann et al. 2016), a blacklisting approach is not a practical solution. This huge number of information discovering C&C connectivity entails a key challenge to address the big data problem (Watkins et al. 2017). In this context, this work focuses on the detection approach, aiming to prevent malwares to contact C&C servers by distinguishing between legitimate and malicious DNs, taking into account big data analysis algorithms for reduction. And, by its own definition, AGDs are dynamically generated as pseudo-random strings or by combination of dictionary words; consequently, it is possible to separate legitimate DNs from the others by applying Machine Learning (ML) techniques. Under the hood, our ML framework leverages two separate families of metrics: on one side static and dynamic analysis of Domain Name Service (DNS) queries highlights suspicious behaviours of clients, while on the other side, natural language processing (NLP) techniques permit to accentuate linguistic differences between DNs. We called these two families Context-Aware and Context-Free, respectively.

Although different in nature, these two families of metrics are complementary. DNS queries analysis can provide evidence of non-human activity, such as daily similarity and repeating communication patterns. For example, it has been proven that legitimate users repeat daily patterns (Bilge et al. 2014). Similarly, NLP-based metrics help to identify non-human activities; that is, DNs are intended to be easily remembered by users, or at least mnemonic, since the main purpose of any DNS service is to provide human-readable association with IP addresses. By comparison, most malware families do not include such forethoughts (Bilge et al. 2014). DNS-related metrics can natively deal with the concept of fast-flux botnets, a subset of the DGA-based family. This technique, spotted in the wild since at least 2007 (with the Storm Worm (Holz et al. 2008), consists in registering multiple IP addresses within the same DNS-A record, leveraging the well-known round-robin DNS scheme to provide short-lived C&C rendezvous-points.

Due to privacy concerns, it is important to state also that metrics based on network analysis are difficult to obtain and use. However, Context-Free metrics are anonymous and more privacy-oriented since they do not require any contextual information from the users or the network state. In other words, users' confidentiality is compromised by packet inspection and network analysis. Additionally, the literature review (Sect. 2) and preliminary results (Sect. 3.3) suggest that these confidentiality-harming solutions may be subdued by more privacy-oriented solutions that do not require users' contextual information in order to achieve excellent detection results.

A key challenge in this field is the comparability of different models. From the theory of ML, it is clear that there are three key aspects of any learning model: data sources, feature characterisation and model optimisation. This work focuses on homogenising the existing Context-Free features in order to be able to test different models with the same data source.

Lastly, it is worth stating that in this research paper two different ML problems are dealt with; that is, given a set of fully qualified domain names (FQDNs), (i) binary separate legitimate domains from malware ones and (ii) categorise them according to their malware family.

Therefore, the contributions of this work are threefold:

- Firstly, a horizontal survey of the state of the art in terms of features used in ML and Deep Learning (DL) approaches to solve cybersecurity challenges related to DGA-based botnets;
- secondly, the analysis of the two aforementioned ML problems and their potential solution using privacy-preserving approaches; and,
- thirdly, a deep analysis of the previously mentioned features to highlight their properties with respect to the feature engineering process and their evaluation using six amongst the most important ML algorithms.

To do so, this article firstly recollects the usage of both families (Context-Aware and Context-Free) in the literature, aiming to establish which features are used, by whom and with which results. Section 2 presents these outcomes. Secondly, in Sect. 3, this work proposes an analysis of the Context-Free features using two different feature selection and extraction techniques and evaluated using six differ-

ent classifiers. Finally, conclusions and the further work are drawn and discussed in Sect. 4.

2 DGA-based botnet detection in the literature

Despite the efforts spent to fight them, DGAs are still thrusting a good portion of the most advanced malware families. As stated before, classic blacklisting approaches are inadequate to contrast malwares, just consider that the DGA powering the now-obsolete Conficker malware can generate up to 50 thousands DNs per day. As illustrated in Kührer et al. (2014), public blacklists were lacking in terms of DGA coverage with less than 1.2% of DGAs analysed by the authors being contained in any of the blacklists. Bruteforcing those generation algorithms is thus not going to solve the problem (especially when considering 0-days variants); however, Artificial Intelligence (AI) can help to tackle them. AI techniques and precisely ML algorithms have been not only proven applicable but also relevant and well suited when tackling these malwares (Tran et al. 2018; Zhang et al. 2016). Along with the classic network detection techniques, e.g. honeypot-based or signature-based, Alieyan et al. (2017) cited passive DNS-based anomaly detection techniques based on Graph Theory (GT), entropy, statistics, Neural Network (NN), Decision Tree (DT) and clustering. Our aim is to extend this taxonomy to provide a more complete and sound classification to achieve better performance when detecting DGA-based botnets; that is, we build a taxonomy based on:

1. The ML approach used, that is either supervised, not supervised or semi-supervised.
2. The families of features adopted in the learning model, that is Context-Aware features (e.g. DNS inspection), Context-Free features [e.g. lexical analysis (Fu et al. 2017] and, finally, a Featureless model [e.g. NN-based learners (Mac et al. 2017)].

To this extent, and with respect to the aforementioned taxonomy categories, this research article will adhere to the following definitions:

Definition 1 (*Context-Aware feature*) A feature that is dependent on the specific malware sample execution, which is realised in a precise environment with a specific configuration and in a particular time frame; for example, Features extracted upon DNS-response inspection.

Definition 2 (*Context-Free feature*) A feature that is related only to a FQDN and thus is independent of contextual information, including, but not limited to, timing, origin or any other environment configuration. First and foremost example of this family is the lexical analysis of the domain name.

In summary, the Context-Free feature family represents the complement set of the Context-Aware feature, that is, a feature can either belong to the Context-Aware or the Context-Free family, but not both.

With regards to the Context-Aware feature family, they are subject to a number of limitations that should be taken into account, so exiting data sets containing them are rare, outdated and generally partial. For example, on the one hand, AGDs lists such as Malware Domain List (2009), Abakumov (2016), Risk Analytics (2007) and OSINT are limited, fragmented and generally outdated. On the other hand, bigger repositories of network traces such as Biglar Beigi et al. (2014) and García et al. (2014) are crafted, heavily unbalanced and often include only short burst of malware packets. They are usually collected in a test environment or handcrafted because mass-collecting real user data are, in fact, a direct breach of user's privacy, and can therefore only be gathered in an anonymous way after explicit consent. Nonetheless, the quality of these data sets is notable (Biglar Beigi et al. 2014; García et al. 2014), and when correctly used, they can be of great help to any detection model. On the contrary, data sets with Context-Free features (Malware Domain List 2009; Abakumov 2016; OSINT; Risk Analytics 2007) consist mainly of lists of FQDNs belonging to a specific malware family. They are natively privacy-oriented since users' data are not involved in any phase of the processing.

Finally, it is worth mentioning that in the literature exist several models that do not make use of hand-crafted, or even automatically guessed features. These models can take advantage of both types of data sets, thus we consider them as a third, distinct category. We define such models "Featureless":

Definition 3 (*Featureless model*) A Machine Learning model that does not require features in order to learn the training data set.

Both Mac et al. (2017), Woodbridge et al. (2016) and Vinayakumar et al. (2018) claim that feature-based detection systems can be easily circumvented by malware engineers in the context of the Adversarial Machine Learning theory. The motivation for such claims generically relies on the intrinsic difficulties of defining and analysing the feature set suitable for not only differentiate AGDs from legitimate FQDNs but also to separate and pinpoint different malware families.

On one side, it is clear that featureless detection systems can produce good results without the need of ideate a feature set; but on the other side, an extensive and deep knowledge about the specific subject represented by the data enables the feature engineering process to converge to an optimal feature set. Feature-based detection systems are in general more reliable and offer important qualities such as the transparency, efficiency, scalability and the capabilities of fine-tuning the

algorithms. Not all features, though, are strictly relevant or helpful during the detection process, i.e. a given feature might not be relevant *as-is* for the detection model, but in combination with others it may produce optimal results.

To reflect the taxonomy we propose in this research article, this section is divided into two subsections according to the ML paradigm chosen by the authors of the different related works of the literature that we have thoroughly analysed, either supervised or unsupervised learning.

Generally speaking, the act of labelling a data set such as a collection of domain names is an extremely complicate and resource-consuming task, especially because white and blacklists can help only to a certain extent. unsupervised learning is, unlike supervised, missing the knowledge related to the instances, i.e. the model does not know the label of each point in the data set. It is imperative to understand that supervised and unsupervised learning are different both in nature and in scope of application: on the one hand, supervised learning techniques partition a data set into subsets, named *clusters*, according to some common characteristics; on the other hand, unsupervised learning algorithms divide the instances into classes and use that knowledge to infer the class (label) of new and previously unseen elements.

With regards to all the aforementioned aspects, in each subsection, a list of the most used or relevant algorithms for that category is presented. Furthermore, each subsection will provide several tables that report:

- the reference of the work;
- the type of classifier or cluster used;
- an indication whenever the authors made a comparison with other works or other methods;
- whether their proposed framework is capable of realtime (RT) detection;
- the algorithm proposed;
- the usage of either Context-Aware or Context-Free features; and
- a generic field that considers the overall results — specifically, we consider as *poor* performances whenever the proposed results (in terms of precision and recall) are below 75%, *average* below 85%, *good* below 95% and *excellent* above 95%.

Remarkably, it is worth noticing that only a few authors have cited any challenge related to 0-day, either as part of their analysis (Nguyen et al. 2015; Pu et al. 2015; Tong and Nguyen 2016) or as potential future work (Ahluwalia et al. 2017; Fu et al. 2017; Thomas and Mohaisen 2014).

2.1 Supervised machine learning approach

A supervised ML algorithm is a function that associates a label (also known as outcome or dependent variable) to a given set of predictors (also known as independent variables). The learning process consists of optimising the internal parameters of the algorithm to associate the input set with the desired output. Examples of supervised ML algorithms are Decision Tree (DTs), Random Forests (RFs), k-Nearest Neighbours (kNN), Hidden Markov Models (HMMs), Neural Network (NNs), etc. Amongst them, we considered only the most effective that have been successfully used in the recent past to detect DGA-based malwares in the network. Here follows a brief list of such approaches.

2.1.1 Hidden Markov Model (HMM)

HMMs are the simplest class of dynamic Bayesian Networks (BNs) and, specifically, they are Markov Models in which the states are hidden (unobservable). HMMs have a proven record of successful applications in the linguistic field when applied to the extraction of grammars and information from texts.

In the literature, they have been used by several authors in order to distinguish legitimate DNs from malicious AGDs. On one side, for example, Mac et al. (2017), Tran et al. (2018), Woodbridge et al. (2016) showed that HMMs behave poorly in comparison with more complex featureless solutions such as Long Short-Term Memory Networks (LSTMs). In general, HMMs have unsatisfactory performances when applied to binary classification between AGDs and legitimate DNs (Woodbridge et al. 2016). HMMs, however, have been successfully deployed by Antonakakis et al. (2012) to trace back the C&C servers, achieving interesting results only when targeting specific classes of malware. Table 1 presents a comparison of these previous works, especially with Deep Learning (DL) and feature aware solutions.

On the other side, however, Fu et al. (2017) decided to use HMMs and probabilistic context-free grammars (PCFGs) to extract core properties of legitimate DNs, in order to build a new family of DGAs able to guarantee the generation of statistically undistinguishable AGDs when referring to lexical analysis (as specified below in Sect. 3). As reported by the authors, the inclusion of others lexical properties, in combination with network-based features, helps in overcoming this concealing technique. Extending the work proposed in Anderson et al. (2016), in terms of using Generative Adversarial Network (GAN) as an anti-detection approach, may result in interesting outcomes. Table 2 presents a comparison of the work used to evade the detection techniques.

Table 1 Supervised approach—Hidden Markov Models

Refs.	Classifier	Comparison	RT	Algorithm	Feature context		Results
					Aware	Free	
Antonakakis et al. (2012)	C&C identification	✗	✗	HMM	Featureless		Poor
Mac et al. (2017) and Tran et al. (2018)	MultiClass	Feature aware, DL	✗	HMM	Featureless		Poor
Woodbridge et al. (2016)	MultiClass	Feature aware, DL	✗	HMM	Featureless		Poor

Table 2 Supervised approach—avoiding detection

Refs.	Used for	Comparison	Algorithm
Anderson et al. (2016)	AGDs generation	Other DGAs	GAN, LSTM
Fu et al. (2017)	AGDs generation	Other DGAs	PCFG, HMM

2.1.2 Artificial neural network (NN) and deep learning

Artificial Neural Networks (NN) are mathematical structures that combine nonlinear functions to compute complex functions. They ultimately aim to resemble the structure of human neurons and interactions. One of the most impressive results of NNs is that it has been proved that they can approximate the result of any function [Universality Theorem (Haykin 1998)]. There is a catch, though, in using NN for such task: they can only approximate continuous functions. Deep Neural Networks come to improve this result.

Amongst the past decades, NNs have been widely and successfully used for image processing, speech recognition and text analysis (Abdel-Hamid et al. 2014). NNs have been adapted to several problems by changing the composition and the number of the inner layers to align with complex problems. As, for example, Baruch and David (2018) designed a NN with a single strongly connected hidden layer and a single output neuron to binary distinguish between legitimate domains and AGDs.

In the case of DGAs, Extreme Learning Machines (ELMs) have been proved effective in the classification of such domains (Mac et al. 2017; Tran et al. 2018; Shi et al. 2017). Nevertheless, NNs have a major shortcoming, the lack of any persistence mechanism. As a result, Recurrent Neural Networks (RNNs), LSTM (more commonly known as Deep Learning) and a bunch of other techniques, including Convolutional Neural Network (CNN) and Cost-Sensitive Neural Network (CS-NN), have been developed to include, respectively, short-term and long-term persistence. Mac et al. (2017) and Tran et al. (2018) studied and developed a specific variation of classic LSTMs to include binary and multiclass classification models with class-dependent cost-sensitive functions. Despite the very good performances in binary classification, they are still unable to distinguish malwares using pronounceable AGDs.

To be more precise, the performances of the most advanced DL techniques (Mac et al. 2017; Tran et al. 2018; Woodbridge et al. 2016) are achieving excellent results only in the binary case, i.e. when distinguishing between malware and legitimate FQDNs. In the multiclass classification, and especially with regards to the most advanced malware families (such as Kraken, CryptoWall or Qakbot), DL-based detection solutions achieve questionable results in terms of both precision and recall.

Woodbridge et al. (2016) also used LSTMs to learn the sequence of patterns generated by DGAs, ultimately classifying AGDs and legitimate DNs. Vinayakumar et al. (2018) also proved the excellent results of RNNs and LSTMs (and their combinations) when solving the binary classification problem of distinguishing legitimate and harmful domain names. Table 3 presents a comparison of the previous works.

As a final remark, it is worth mentioning that DL has many issues that are usually skipped during the evaluation phase of the proposed works. Although it is correct that they can offer impressive results, it is also correct that they are often overfitted and especially opaque. This lack of transparency ultimately leads to the impossibility of fine-tuning the algorithms and of explicating the reasons behind the results. As for HMMs, Deep Learning can be used to outshine the concealing capabilities of classic DGAs. It is the case studied and reported by Anderson et al. (2016), who developed a DGA specifically designed for crafting difficult DNs. The AGDs were used then to harden the proposed classifier, resulting in a GAN architecture able to lower the detection rate below any acceptable threshold. Related works used to evade the detection techniques have been presented and compared above in Table 2.

2.1.3 Decision trees (DTs) and derived

Contrarily to the Deep Learning solutions presented in Sect. 2.1.2, Decision Trees (DTs) offer transparent solutions that do not require any scaling or data normalisation. Moreover, since they are particularly resilient to outliers, missing values and nonlinear relationships, they are capa-

Table 3 Supervised approach—neural network and deep learning

Refs.	Classifier	Comparison	RT	Algorithm		Feature context		Results
				NN-based	LSTM	Aware	Free	
Baruch and David (2018)	Binary	Feature aware	✗	NN	✗	✗	✓	Good
Mac et al. (2017) and Tran et al. (2018)	MultiClass	HMM, feature aware	✗	CS-NN, ELM	CNN-LSTM	Featureless		Excellent[a]
Shi et al. (2017)	Binary	Feature aware	✗	ELM	✗	✓	✓	Excellent
Vinayakumar et al. (2018)	MultiClass	Feature aware	✗	I-RNN, CNN	CNN-LSTM	Featureless		Excellent
Woodbridge et al. (2016)	MultiClass	HMM, feature Aware	✗	✗	✓	Featureless		Excellent

[a]While classifying some classes. The scores are excellent when considering micro-averaging the per-class scores, but on macro-averaging the results are quite poor

ble of remaining consistent regardless of the data shape. Nonetheless, they are prone to errors and misconfiguration, i.e. without pruning and limitations they tend to overt the data, thus lowering the prediction accuracy. Although it is common to find a scenario-specific algorithm that performs better [typically DL or AdaBoost (AB)], they conventionally require more resources or time to be trained.

Several works have been proposed so far in the context of DTs, not surprisingly using the classic C4.5 as generation algorithm. Specifically, Ahluwalia et al. (2017) and Bilge et al. (2014) have used them to solve the binary classification problem of distinguishing AGDs from FQDNs, while Antonakakis et al. (2012), Mac et al. (2017), Stevanovic et al. (2017), Tran et al. (2018), Truong and Cheng (2016), Vinayakumar et al. (2018) have instead opted for the multi-classification problem of identifying the malware family.

Random Forests (RFs) have been proposed to help to fix the aforementioned overfitting bias of DTs. RFs require almost no parameter tuning, as DTs can handle any type of feature without scaling or normalisation. They do not require tweaking of the hyper-parameters, perform implicit features selection and train extremely fast. However, these accomplishments are often paid with the slow evaluation performances and the important amounts of resources required creating and storing them. Nevertheless, RF, out of the box, performs particularly well. Not surprisingly RFs have been widely used in the literature to approach the DGA-based botnet problem both in the binary (Ahluwalia et al. 2017; Xu et al. 2017) and in the multiclass (Luo et al. 2017; Song and Li 2016; Stevanovic et al. 2017; Truong and Cheng 2016; Vinayakumar et al. 2018; Woodbridge et al. 2016) forms.

As specified before, both DT-based and RF-based models need features in order to work. Generally speaking, the usage of Context-Free features (as defined in Sect. 2) is sufficient to have a good-to-excellent classifier. To be more precise though, it is worth mentioning that to the best of our knowledge, there are not related researches that only uses Context-Aware features for the classification with DT or RF algorithms. In fact, when considering Context-Aware metrics they tend to have them integrated with NLP-based ones (Bilge et al. 2014; Stevanovic et al. 2017; Xu et al. 2017), while on the contrary, researches featuring DT and/or RF solutions tend to focus purely on linguistic features (Ahluwalia et al. 2017; Luo et al. 2017; Mac et al. 2017; Song and Li 2016; Tran et al. 2018; Truong and Cheng 2016; Xu et al. 2017).

Table 4 presents a comparison of these previous works.

It may perhaps be observed that the authors have used a different subset of these features, both belonging to the Context-Aware and Context-Free families. To clarify this observation, Sect. 3.2 will highlight and present, to the best of our knowledge, the complete list and description of the used features.

It also worth mentioning how Vinayakumar et al. (2018) used a CNN to generate the features that later on have been analysed by the DT and the RF classifier. Moreover, highlight how these classifiers are compared with several algorithms that are intrinsically different, including classic approaches such as SVM, NB and SGD.

2.1.4 Other supervised approaches

Apart from DL and DT derived learners, historically exist several other decision algorithms. Amongst them, the five most used algorithms are presented in Table 5 and described in the following list.

- *Naïve Bayes (NB)*, a family of probabilistic classifiers that assume a strong (and thus naïve) independence between the features. In the context of AGDs detection, it has been applied with inconsistent results (Truong and Cheng 2016; Vinayakumar et al. 2018).
- *Regression*, both linear and logistic regressions represent a family of learners that attempt to define an explanation model by interpolating the data. To the best of our knowledge, this approach has not yet been studied and applied to the problem.

Table 4 Supervised approach—decision trees and derived

Refs.	Classifier	Comparison	RT	Algorithm		Feature context		Results
				DT	RF	Aware	Free	
Ahluwalia et al. (2017)	Binary	✗	✗	✓	✓	✗	✓	Excellent
Antonakakis et al. (2012)	MultiClass	✗	✗	✓	✗	✗	✓	Excellent
Bilge et al. (2014)	Binary	✗	✓	✓	✗	✓	✓	Excellent
Luo et al. (2017)	MultiClass	✗	✓	✗	✓	✗	✓	Good
Mac et al. (2017)and Tran et al. (2018)	MultiClass	SVM, DL	✗	✓	✗	✗	✓	Average
Song and Li (2016)	MultiClass	✗	✗	✗	✓	✗	✓	Good
Stevanovic et al. (2017)	MultiClass	✗	✓	✓	✓	✗	✓	Good
Truong and Cheng (2016)	MultiClass	NB, kNN, SVM	✗	✓	✓	✗	✓	Good
Vinayakumar et al. (2018)	MultiClass	AB, NB, DL	✗	✓	✓	CNN-generated		Excellent
Woodbridge et al. (2016)	MultiClass	HMM, DL	✗	✗	✓	✗	✓	Excellent
Xu et al. (2017)	Binary	AB, SGD	✓	✗	✓	✓	✓	Excellent

Table 5 Supervised approach—other machine learning techniques

Refs.	Classifier	Comparison	RT	Algorithm	Feature context		Results
					Aware	Free	
Baruch and David (2018)	Anomaly	DL	✗	SVM, kNN	✗	✓	Good
Han and Zhang (2017)	Binary	Accuracy (Luo et al. 2017)	✗	SVM	Filter	✓	Good
Mac et al. (2017) and Tran et al. (2018)	MultiClass	DT, RF, DL	✗	SVM	✗	✓	Good
Truong and Cheng (2016)	MultiClass	DT, RF	✗	NB, kNN, SVM, AB	✗	✓	Poor
Vinayakumar et al. (2018)	MultiClass	RF, DT, DL	✗	NB, AB	CNN-generated		Excellent
Xu et al. (2017)	Binary	DT, RF	✗	SGD, AB	✓	✓	Excellent

– *Support Vector Machines (SVMs)* represent a family of linear models that attempt to classify data by finding a hyperplane that maximises the data distance in an n-dimensional space. A common approach is to use the SVM as binary classifier to separate legitimate and malicious AGDs (Baruch and David 2018; Han and Zhang 2017), but also as multiclass classifier (Mac et al. 2017; Tran et al. 2018; Truong and Cheng 2016).
– *k-Nearest Neighbours (kNNs)*, a classifier that defines the boundaries of the classes by the distance from their neighbours through a majority vote. In one case, Truong and Cheng (2016) successfully applied this algorithm to the problem, obtaining poor results. However, Baruch and David (2018) have used this approach for anomaly detection, obtaining interesting results.
– *Stochastic Gradient Descend (SGD)*, a family of classification algorithms that approximate gradient optimisation. Xu et al. (2017) have reportedly used it for comparison.

As specified before, and similarly to the previous subsections, Table 5 reports the comparison of these works in terms of RT usage, algorithms, feature families and results.

2.2 Unsupervised machine learning approach

Amongst the algorithms making use of unsupervised learning techniques, it appears that in the literature the K-Means, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and Hierarchical Clustering (HC) algorithms are the most used. To this extent, it is clear that unsupervised learning has a concrete and important advantage compared with supervised learning, as it does not require labelled data sets. However, the labels of each cluster's instance can be inferred by a few manually recognised examples. The main drawback of most of unsupervised learning algorithms resides in the fact that they need to be configured with the predicted number of clusters, an assumption that may not be available.

In the literature, unsupervised learning has been applied in many scenarios with different objectives, including, but not limited to:

– statistical filtering techniques to filter out the most basic AGDs (Grill et al. 2015);
– techniques to correlate bots and C&Cs (Han and Zhang 2017);

Table 6 Unsupervised approach—K-Means and derived

Refs.	Cluster	Comparison	RT	Algorithm		Feature context		Results
				K-Means	Other	Aware	Free	
Antonakakis et al. (2012)	Multiclass	✗	✗	X-Means	✗	✗	✓	Excellent
Bisio et al. (2017)	Multiclass	✗	✓	✓	✗	Filter	✓	Unclear[a]
Nguyen et al. (2015)	Multiclass	DBSCAN	✗	X-Means	✗	✓	✓	Average
Pu et al. (2015)	Multiclass	✗	✗	✓	✗	✗	✓	Unknown
Stevanovic et al. (2015)	Binary	✗	✗	✓	✗	✓	✓	Good
Tong and Nguyen (2016)	Multiclass	DBSCAN	✗	X-Means	✗	Filter	✓	Unclear[a]

[a] Authors are not providing detection metrics such as accuracy, precision and recall

– techniques to label groups of FQDNs according to their similarity (Berger and Gansterer 2013; Stevanovic et al. 2015);
– detection tools to separate legitimate from malicious FQDNs (Baruch and David 2018; Pu et al. 2015); and
– techniques to group users according to their network behaviour, e.g. users who query the same group of non-existent domains (NXDomains) (Antonakakis et al. 2012).

A clear trend in the literature is the usage of unsupervised learning techniques to perform some sort of data preprocessing prior to performing real classification with more resource-consuming supervised algorithms. Several works are, in fact, at least partially based on clustering and/or association, such as the ones presented in Antonakakis et al. (2012), Fu et al. (2017) and Nguyen et al. (2015), just to reference a few.

2.2.1 K-means and derived

The well-known K-Means algorithm is the first and the simplest unsupervised learning algorithm. It requires defining *a priori* the number of desired clusters and maps each point in the data set with the same label of the nearest cluster centroid. These centroids evolve and move during the iterations of the algorithm, ensuring the adaptability of the algorithm to different scenarios and applications. It is worth noticing that although it produces an output in a finite amount of time, it may not be the optimal result, which may be strongly dependent from the initial configuration. Essentially, it has three major drawbacks (Pelleg and Moore 2000), that is, it scales poorly, the number of clusters has to be supplied beforehand and it may output local optimal results.

Several authors Bisio et al. (2017), Pu et al. (2015), Stevanovic et al. (2015) have used the K-Means to, at least partially, solve the AGDs detection problem. Knowing the aforementioned issues, Antonakakis et al. (2012), Bisio et al. (2017), Tong and Nguyen (2016) have instead opted for a variant known as X-Means (Pelleg and Moore 2000) that estimates the number of clusters and their parameters, facing though a sensible amount of resources required (Nguyen et al. 2015). Table 6 presents a comprehensive list of literary works that have been reportedly used either this algorithm or a derived version.

Using an appropriate feature set, the K-Means algorithm is capable of associating collected FQDNs in clusters, which can be later on used to recognise 0-day AGDs with similar characteristics to previously discovered malwares (Tong and Nguyen 2016). Such module can be used as a filter to collect and then discard the AGDs generated by known malwares, in order to focus on the ones that are potentially generated by new DGAs.

2.2.2 Other unsupervised approaches

Amongst the historically notable unsupervised learning algorithms, it appears, to the best of our knowledge, that only a few of them have been studied and presented in the past decade to deal with the problem of detecting DGA-based botnets. In fact, apart from the K-Means and its variants highlighted in the previous section, there are two approaches to clustering: the Mixture Model (MM) and the HC, but their usage is quite limited and with questionable results.

To be more precise, the MM attempts to model the data by using a mixture of probability distributions, while the HC uses distance (and linkage) functions to separate (or join) groups of points in the data set. While the HC has been proposed a few times (Zhang et al. 2016; Thomas and Mohaisen 2014; Tu et al. 2015; Fu et al. 2017), to the best of our knowledge the MM clustering has not been applied so far to this subject. Similarly, the Bipartite Graph Clusterings (BGCs) approach used by Han and Zhang (2017) does not need to indicate a number of clusters and it has the ability to find any geometric clusters. But, in the course of running, this algorithm requires high computational resources when running online in comparison with K-Means (Tong and Nguyen 2016).

Finally, we can also find the DBSCAN algorithm that, similarly to the agglomerative HC approach, join together

Table 7 Unsupervised approach—other algorithms

Refs.	Usage	Comparison	RT	Algorithm	Feature family		Results
					DNS	NLP	
Han and Zhang (2017)	C&C identification	✗	✗	BGC	✓	✗	Unknown
Zhang et al. (2016)	Detection	✗	✗	HC	✗	✓	Average
Thomas and Mohaisen (2014)	Detection	✗	✗	HC	✗	✓	Average
Tu et al. (2015)	Correlation	✗	✗	HC	✓	✗	Good
Fu et al. (2017)	Correlation	✗	✗	HC	✗	✓	Average
Nguyen et al. (2015)	Identification	Schiavoni et al. (2014)	✗	DBSCAN	✓	✓	Average
Tong and Nguyen (2016)	Identification	K-Means	✗	DBSCAN	Filter	✓	Average

elements of the data set according to the density of the region in which they reside, being capable of modelling clusters of any spatial shape. A few authors Nguyen et al. (2015) and Tong and Nguyen (2016) have used it in order to circumvent the need of initialising the parameters of the aforementioned K-Means. Table 7 presents a list of these works that have just been discussed.

2.3 Discussion and key points

The proposed state of the art has highlighted a few important inconsistencies in terms of solutions for tackling DGA-based botnets. The foremost notable shortage is undoubtedly the lack of structured data sources, especially when considering those suitable for ML algorithms. The preprocessing phase for the *raw-data* such as the PCAP files or the network flows is not trivial, and its consequent exploratory analysis is considerably time-consuming. As a result, the evaluation of the proposed classifying and clustering algorithms is quite a challenge. Several algorithms have been proposed, and authors have reported discordant results. As previously cited, a common ground may lead to improved and, most importantly, comparable results.

For another thing, although most of the reported works are focusing on the multiclass analysis of malware families, the binary case should not be *a priori* excluded. Further researches are required in order to establish a signature for the legitimate FQDNs so to be able to filter out suspicious DNS queries for subsequent analysis.

Finally, it is worth noticing the staggering absence of solutions capable of targeting 0-day malware variants and families. In fact, only a few authors Nguyen et al. (2015), Pu et al. (2015), Tong and Nguyen (2016) have included at least a partial analysis of the subject, even if, intuitively, this may be explained through the native predisposition of unsupervised learning techniques towards unknown samples and classes. In brief, the 0-day detection still represents an open research topic in detection as well as in other phases of the cybersecurity process (Lobato et al. 2018; Nespoli et al. 2018).

To this extent, we define the following challenges and potential research lines that should be considered by the cybersecurity community and industry.

1. Firstly, privacy-oriented data sets must be researched and made publicly available;
2. secondly, it is mandatory to establish a series of shared best practices that lead and advise future researches related to ML applications for botnet detection;
3. thirdly, to enable the research community to focus on the study of new approaches and detection algorithms instead of data gathering and preprocessing, ML-oriented and ready-to-use data sets must be researched and made publicly available; and,
4. finally, having taken into consideration the delicate and complex nature of the data, *ad-hoc* nonlinear solutions might be worth investigating.

To summarise, the scientific community, but also the security vendors, might benefit from exploiting the aforementioned research lines. Ideally, by establishing a common ground in terms of data, feature sets, procedures and eventually reactions, the future researches might only focus on providing algorithms and advanced solutions for detection, reaction and mitigation purposes.

3 Defining a common base for feature analysis

In Machine Learning, the process of feature analysis is well known for being complex and extremely time-consuming (Bishop 2006; Almomani et al. 2018), conjecture that has been confirmed once again in the case of DGA-based botnet detection (Woodbridge et al. 2016). It also requires a deep knowledge of the data in conjunction with the algorithm that attempt to model the training data. In this context,

and as specified in Sect. 2, we consider two macro-categories for features related to DGA-based botnet detection, namely Context-Aware (Definition 2) and Context-Free (Definition 1) features. They are, respectively, dependent on the instantiation of the malware sample (e.g. DNS response Time-to-Live, TTL) and independent from it (e.g. the number of characters in the queried FQDN). Clearly, being the focus of this work on feature analysis, we *a priori* exclude the Featureless (Definition 3) solutions. Similarly, we exclude the automatic generated features solutions due to their strong dependence from the input data and not on the definition of the features itself.

This work only focuses on the Context-Free category that, being independent of the state of the network, permits us an evaluation not susceptible to the environment variations occurring where and when the malware is executed. As a consequence, further researches are required in order to analyse the Context-Aware feature family.

The features collected and presented in this work are *individual*; they are extracted from a single and precise FQDN (e.g. the domain length) so that the feature set does not include *aggregated* features such as the average of such domain length.

This section is divided into three parts. Section 3.1 briefly introduces the methodology and the data used for carrying out the comparison, Sect. 3.2 highlights the most informative features existing in the literature and, finally, Sect. 3.3 presents a detailed evaluation of these features through a series of experiments.

3.1 Methodology

In order to be as variate as possible, we retrieved a list of collected AGDs from public available data sets (Bader; Netlab 360; Plohmann 2015; Risk Analytics 2007) and malware blacklists (Malware Domain List 2009; OSINT). The following families using DGAs were taken as data source due to their importance, usage or complete reverse engineering status: Alureon, Conficker, CryptoLocker, Goz, Kraken, Matsnu, Murofet, Nymaim, Pushdo, QakBot, Ramdo, Rovnix, Shiotob, Simda, Tinba and Zeus. To establish a comparison with the real-world legitimate FQDNs, we retrieved the list of the top-ranking domains according to Majestic-12 Ltd: The Majestic Million (2018) backlink data set. Our collected data set includes a thousand distinct samples for each class, with the ones that we consider *a priori* legitimate. We extracted from each sample the list of features described in Sect. 3.2 and, based on the histogram analysis, performed the suggested data preprocessing operations, e.g. transformation, scaling and normalisation. The data have been preprocessed with alternatively Orange3 (Demšar et al. 2013) and Weka (Fran et al. 2016).

In order to do so, we will:

1. Present the list of features extracted from the state of the art in the category of individual Context-Free features.
2. Present a selection of distribution histograms and the correlation matrix from the new feature set obtained from the ranking method.
3. Apply the Principal Component Analysis (PCA) to extract n features that cumulatively cover at least 98% of the variance on the original feature set.
4. Evaluate the original data with the six most used classifiers in the state of the art (RF, NN, SVM, DT, AB and kNN), using the well-known accuracy, precision, recall, area under the curve (AUC) and F1 scores.

3.2 Most informative features

By reviewing the literature on the subject of DGA-based botnet discovery, we collected a total amount of 40 features, of which 17 are related to the NLP nGrams, and thus recalculated for every value of $n \in \{1, 2, 3\}$. Table 8 reports these findings. Specifically, the first 23 features,[1] presented in Table 8 are related to the metrics that can be extracted by analysing the domain name as if it were a string of text. It makes sense, as a consequence, to measure metrics such as its entropy (that measure the randomness of the string), its length and the length of the longest consecutive consonant sequence. Moreover, most of these features reflect real-world common practices such as the Search Engine Optimisation (SEO) and the netiquette. For example, SEO advice includes not only the ideal length of a domain name, which should be around 12–13 characters, but also suggestions like the reading easiness and the ability to spell the relay the FQDN to someone else.

The presented features in Table 8 can be assigned to two groups, on the one side classic string metrics such as the length (67% of cited works) (Schales et al. 2016; Ahluwalia et al. 2017; Antonakakis et al. 2012; Bisio et al. 2017; Han and Zhang 2017; Luo et al. 2017; Mowbray and Hagen 2014; Plohmann et al. 2016; Pu et al. 2015; Schiavoni et al. 2014; Shi et al. 2017; Song and Li 2016; Stevanovic et al. 2017; Tran et al. 2018; Truong and Cheng 2016; Xu et al. 2017), the number of vowels characters (17%) Schales et al. (2016), Ahluwalia et al. (2017), Song and Li (2016), Stevanovic et al. (2017) or the entropy (46%) Antonakakis et al. (2012), Bisio et al. (2017), Han and Zhang (2017), Luo et al. (2017), Plohmann et al. (2016), Pu et al. (2015), Shi et al. (2017),

[1] Including four features (`NLP-L-x`, `NLP-R-NUM-x`, `NLP-R-VOW-x`, `NLP-R-CON-x`) for each domain name level: the FQDN, the Second Level Domain Name (2LD) or all the others sub-levels as a whole (OLD).

Table 8 Context-Free features collected from the literature

Code	Description
(a) Context-Free features with their descriptions	
NLP-L-*x*	String length
NLP-LDN	Number of domain levels
NLP-R-NUM-*x*	Ratio of numerical characters
NLP-R-VOW-*x*	Ratio of vowel characters
NLP-R-CON-*x*	Ratio of consonants characters
NLP-LANG	Language hypothesis
NLP-LC-C	Longest consecutive cons. sequence
NLP-LC-V	Longest consecutive vowel sequence
NLP-LC-D	Longest consecutive number sequence
NLP-COV	Covariance matrix
NLP-R-MC	Ratio of meaningful characters[a]
NLP-LMS	Length of longest meaningful string[a]
NLP-WLU	Number of "word-like" units[a]
NLP-SQS	Domain squatting score[a]
NLP-LED	Levenshtein edit distance[a]
NLP-*n*G-FR	Frequency distribution (histogram)
NLP-*n*G-E	Entropy
NLP-*n*G-COV	Covariance[a]
NLP-*n*G-MEAN	Mean of frequencies
NLP-*n*G-MED	Median of frequencies
NLP-*n*G-VAR	Variance of frequencies
NLP-*n*G-STD	Standard deviation of frequencies
NLP-*n*G-PRO	Pronounceability score[a]
NLP-*n*G-NORM	Normality score
NLP-*n*G-PRT	Transition probability[a]
NLP-*n*G-PRA	Probability of appearance[a]
NLP-*n*G-PRI	Index probability[a]
NLP-*n*G-DST-KL	Kullback–Leiber divergence[a]
NLP-*n*G-DST-JI	Jaccard Index measure[a]
NLP-*n*G-DST-TH	Distance-threshold[a]
NLP-*n*G-DST-AF	Distance–avg. frequency[a]
NLP-*n*G-DST-AC	Distance–avg. count[a]

Used by	Code (NLP-)													
	L-*x*	LDN	R-NUM-*x*	R-VOW-*x*	R-CON-*x*	LANG	LC-C	LC-V	LC-D	COV	R-MC	LMS	WLU	SQS	LED
(b) Context-Free features and their usage in the literature															
Abbink and Doerr (2017)															
Ahluwalia et al. (2017)	✓		✓	✓	✓										
Antonakakis et al. (2012)	✓	✓													
Baruch and David (2018)															✓
Bilge et al. (2014)			✓									✓			
Bisio et al. (2017)	✓	✓													
Han and Zhang (2017)	✓		✓		✓		✓		✓						
Kintis et al. (2017)														✓	

Table 8 continued

Used by	Code (NLP-)														
	L-*x*	LDN	R-NUM-*x*	R-VOW-*x*	R-CON-*x*	LANG	LC-C	LC-V	LC-D	COV	R-MC	LMS	WLU	SQS	LED
Luo et al. (2017)	✓						✓								
Mac et al. (2017)															
Mowbray and Hagen (2014)	✓														
Plohmann et al. (2016)	✓														
Pu et al. (2015)	✓						✓	✓	✓						
Schales et al. (2016)	✓	✓		✓	✓	✓									
Schiavoni et al. (2014)	✓		✓							✓	✓				
Shi et al. (2017)	✓						✓								

Used by	Code (NLP-)								
	*n*G-FR	*n*G-E	*n*G-COV	*n*G-MEAN	*n*G-MED	*n*G-VAR	*n*G-STD	*n*G-PRO	*n*G-NORM
(b) Context-Free features and their usage in the literature									
Abbink and Doerr (2017)									
Ahluwalia et al. (2017)									
Antonakakis et al. (2012)	✓	✓			✓	✓	✓		
Baruch and David (2018)									
Bilge et al. (2014)									
Bisio et al. (2017)		✓							
Han and Zhang (2017)		✓						✓	
Kintis et al. (2017)									
Luo et al. (2017)	✓	✓							
Mac et al. (2017)									✓
Mowbray and Hagen (2014)									
Plohmann et al. (2016)		✓							
Pu et al. (2015)	✓	✓							
Schales et al. (2016)									
Schiavoni et al. (2014)			✓	✓				✓	
Shi et al. (2017)		✓							
Song and Li (2016)		✓						✓	✓
Stevanovic et al. (2017)									
Thomas and Mohaisen (2014)									
Tong and Nguyen (2016)									✓
Tran et al. (2018)		✓							
Truong and Cheng (2016)		✓							
Xu et al. (2017)	✓	✓							
Yadav et al. (2010)									

Used by	Code (NLP-)							
	*n*G-PRT	*n*G-PRA	*n*G-PRI	*n*G-DST-KL	*n*G-DST-JI	*n*G-DST-TH	*n*G-DST-AF	*n*G-DST-AC
(b) Context-Free features and their usage in the literature								
Abbink and Doerr (2017)					✓			
Ahluwalia et al. (2017)								
Antonakakis et al. (2012)								
Baruch and David (2018)				✓	✓			
Bilge et al. (2014)								

Table 8 continued

Used by	Code (NLP-)							
	*n*G-PRT	*n*G-PRA	*n*G-PRI	*n*G-DST-KL	*n*G-DST-JI	*n*G-DST-TH	*n*G-DST-AF	*n*G-DST-AC
Bisio et al. (2017)								✓
Han and Zhang (2017)								
Kintis et al. (2017)								
Luo et al. (2017)	✓	✓	✓				✓	✓
Mac et al. (2017)								
Mowbray and Hagen (2014)								
Plohmann et al. (2016)								
Pu et al. (2015)				✓				
Schales et al. (2016)								
Schiavoni et al. (2014)						✓		
Shi et al. (2017)								
Song and Li (2016)								
Stevanovic et al. (2017)								
Thomas and Mohaisen (2014)					✓			
Tong and Nguyen (2016)								
Tran et al. (2018)								
Truong and Cheng (2016)		✓						
Xu et al. (2017)	✓							
Yadav et al. (2010)			✓		✓			

$x \in$ {FQDN, 2LD, OLD} denotes the domain levels [a] is about the English language $n \in [1, 2, 3]$ represents the n Gram size

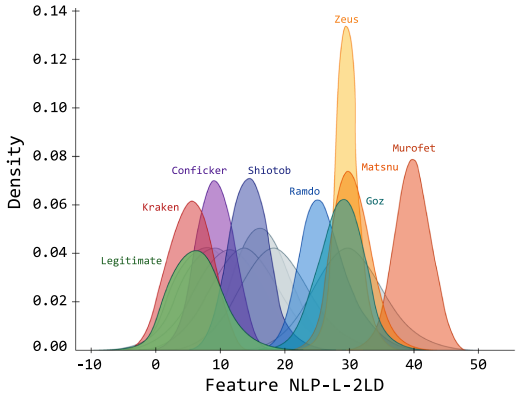
Song and Li (2016), Tran et al. (2018), Truong and Cheng (2016), Xu et al. (2017) shows attempts of solving the problem by using simpler but effective metrics; on the other side, features like the Jaccard Index measure (17%) (Abbink and Doerr 2017; Baruch and David 2018; Thomas and Mohaisen 2014; Yadav et al. 2010), the Kullback–Leiber divergence (0.08%) (Baruch and David 2018; Pu et al. 2015) or the probability of appearance (0.08%) (Luo et al. 2017; Truong and Cheng 2016) depict a deeper knowledge about the structure of the domain names and their usage from the linguistic point of view. Most importantly, however, is the fact that a standard pool of features is missing; that is, most of the presented solutions use arbitrary combinations of them, often with different names and unconventional mathematical definitions.

This work aims to homogenise the features by implementing them according to their theoretical definitions and common usage. Although with different formulations, authors like Ahluwalia et al. (2017) and Schales et al. (2016) have both proposed at a feature based on differentiating between vowels and consonants. Specifically, Ahluwalia et al. (2017) proposed to use the count of the occurrences of the consonants as feature, while Schales et al. (2016) proposed a Boolean flag that indicates whenever the domain name is composed only by consonants. Intuitively, the former is a discrete number whose value can range[2] from 3 to 255 characters that can be easily normalised with respect to the domain length. The latter, however, delineate a partial information by discarding data regarding the string composition. Both features are homogenised in our proposal by the NLP-R-CON-FQDN feature, which measures the ratio between the consonants and the length of the FQDN: on the one hand, it perfectly represents the number of occurrences normalised with respect to the length of the domain, while on the other hand not only includes the Boolean situation where the domain name is composed by all consonants but also provides additional information regarding the domain structure.

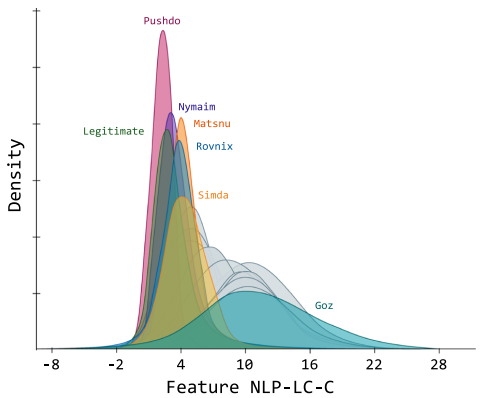
Likewise, a similar process has been completed for all the others features presented in Table 8; however, due to space and complexity concerns, such homogenisation process is not reported in this research item.

We included in Table 8b previous researches that are not strictly aligned with the DGA-based botnet detection, being either grouping client based on their connections (Schales et al. 2016; Yadav et al. 2010), applying filtering techniques (Abbink and Doerr 2017; Mowbray and Hagen

[2] According to ICANN specifics, the minimum length of a domain name without considering the Top Level Domain (TLD) is three characters. The maximum, including symbols and extensions, is 255, having a maximum length *per-level* of 63 characters.



(a) Distribution histogram of NLP-L-2LD.



(b) Distribution histogram of NLP-LC-C.

Fig. 1 Distribution histograms of two selected features from Table 8a

2014) or targeting a different problem (Kintis et al. 2017). Nonetheless, their feature definition and usage are clear and well used.

In order to realise a potential common data set of AGDs suitable for ML algorithms, the data are required to be normalised. To do so, each feature's histogram and relative distribution have been analysed and accordingly transformed to obtain, where possible, a normal distribution. Additionally, their graphical representations permit revealing some important aspects; for example, amongst all the features, we picked two normalised distributions from Table 8, namely `NLP-L-2LD` (length of the second level domain name) and `NLP-LC-C` (longest consecutive consonants sequence).

These features have been chosen as example to illustrate an important property that can be extracted from the distribution analysis, that is, a feature can be used to discriminate according to its value (Fig. 1a) or its shape (Fig. 1b). In fact, it is important to notice that malwares have different and well-defined distributions; for example, in the case of the feature `NLP-L-2LD` (Fig. 1a) the density histogram shows how different values of the feature permits to sort AGDs into their classes. The feature `NLP-LC-C`, however, presents a more overlapped distribution due to its nature (only a few malwares, such as Pushdo and Nymaim have an indistinguishable distribution with respect to the legitimate ones). Yet, its information is enough to separate them by exclusion from the simpler ones.

As for the features related to the *n*Grams, presented in Table 8, and similarly to the aforementioned habits, it is worth mentioning the pronounceability and normality scores, the transition and index probability and the different distances and divergences from the English language. These values are often calculated differently but having the same objective in mind, i.e. establishing "how random" are the domain names. However, they have been accordingly analysed, as illustrated, for example, for feature `NLP-R-CON-FQDN`, and reduced to a common formal definition. In order to analyse this feature

Table 9 Most informative features

Code	IG	IG ratio	Gini	χ^2
(a) Sorted by IG				
`NLP-L-2LD`	1.450	0.725	0.112	13296.997
`NLP-L-FQDN`	1.425	0.716	0.112	14143.591
`NLP-1G-PRO`	1.392	0.697	0.109	12441.981
`NLP-2G-PRO`	1.304	0.652	0.101	12291.421
`NLP-1G-MEAN`	1.000	0.500	0.081	8492.071
(b) Sorted by IG ratio				
`NLP-R-NUM-OLD`	0.306	0.821	0.044	1023.027
`NLP-LC-D`	0.306	0.820	0.044	13236.135
`NLP-L-2LD`	1.450	0.725	0.112	13296.997
`NLP-L-FQDN`	1.425	0.716	0.112	14143.591
`NLP-R-NUM-FQDN`	0.645	0.711	0.069	30694.025
(c) Sorted by Gini Coefficient				
`NLP-L-2LD`	1.450	0.725	0.112	13296.997
`NLP-L-FQDN`	1.425	0.716	0.112	14143.591
`NLP-1G-PRO`	1.392	0.697	0.109	12441.981
`NLP-2G-PRO`	1.304	0.652	0.101	12291.421
`NLP-1G-MEAN`	1.000	0.500	0.081	8492.071
(d) Sorted by χ^2				
`NLP-R-NUM-FQDN`	0.645	0.711	0.069	30694.025
`NLP-R-NUM-2LD`	0.634	0.699	0.065	30031.921
`NLP-L-FQDN`	1.425	0.716	0.112	14143.591
`NLP-L-2LD`	1.450	0.725	0.112	13296.997
`NLP-LC-D`	0.306	0.820	0.044	13236.135

Table 10 Correlation matrix of the most informative features

Features (NLP-)	R-NUM-FQDN	L-FQDN	L-2LD	LC-D	1G-PRO	2G-PRO	1G-MEAN	R-NUM-OLD	R-NUM-2LD
R-NUM-FQDN	–	0.51	0.51	−0.11	−0.14	0.04	−0.04	0.02	0.99
L-FQDN	0.51	–	0.96	−0.1	−0.81	−0.59	−0.16	−0.01	0.47
L-2LD	0.51	0.96	–	−0.36	−0.76	−0.55	−0.14	0.00	0.47
LC-D	−0.11	−0.10	−0.36	–	0.03	0.03	−0.01	0.00	−0.11
1G-PRO	−0.14	−0.81	−0.76	0.03	–	0.86	0.21	0.03	−0.08
2G-PRO	0.04	−0.59	−0.55	0.03	0.86	–	0.19	0.05	0.08
1G-MEAN	−0.04	−0.16	−0.14	−0.01	0.21	0.19	–	−0.01	−0.02
R-NUM-OLD	0.02	−0.01	0.00	0.00	0.03	0.05	−0.01	–	0.00
R-NUM-2LD	0.99	0.47	0.47	−0.11	−0.08	0.08	−0.02	0.00	–

set, we extracted the five most relevant features according to four different algorithms, reported in Table 9; namely:

- *Information Gain (IG)*, sorted in Table 9a. It gives a score regarding "how much" information is the feature bringing with respect to the classification target.[3] Substantially, it reflects the amount of information that was available before and after considering the feature.
- *IG Ratio*, sorted in Table 9b. Similar in concept to the IG, this score measures the ratio between the information provided by the actual label and the one provided by the feature. It compensates for high-entropy features, which are normally advantaged by the simple IG, score.
- *Gini Coefficient*, sorted in Table 9c. This metric measures the "purity" of the feature with regards to the actual label. The greater the value is, the better the feature is.
- *Chi-Square* (χ^2), sorted in Table 9d. This statistical score measures the independence (specifically its lack) between the feature and the actual label compared with the χ^2 distribution with one degree of freedom. The greater the value is, the more category information the feature contains.

As the features' names suggest, the length of the domain as a whole and the length of the second level domain name might be strictly correlated, as well as the number ratio for the second and the FQDN. We thus extracted the correlation matrix, presented in Table 10, which confirms our hypothesis by identifying a strong correlation between two different pairs of features.

Generally speaking, having highly correlated features affects the model performances, especially in supervised learning. However, to be more precise, both the improvement and the deterioration in model performances are questionable and require further investigations. Removing them may be the correct approach, but also may degrade the model:

- *Performance improvement*. Especially in terms of training speed, due to the well-known problem of the "Curse of Dimensionality".
- *Bias decrease*. As a rule of thumb, features that present low values of mutual correlation or multicollinearity, to the target are helpful and generally speaking they should be kept in the feature set.
- *Interpretability*. Fewer features lead to a model that is easier to justify and explain, thus it may worth to remove them, possibly paying in terms of precision and recall.

Within the context of our data set and the aforementioned applicability scenario, we decided to discard the feature NLP-L-2LD in favour of NLP-L-FQDN since former is stripped from the extensions, thus losing information with respect to the latter. Likewise, we discard the NLP-R-NUM-FQDN in favour of NLP-R-NUM-2LD due to the ICANN rules that prevent numerical characters from appearing in the extension part.

Furthermore, feature engineering algorithm such as the PCA may provide another hint towards a more suitable feature set. Specifically, the PCA decorrelates the data by mapping it to n orthogonal dimensions that maximise the variance and minimise the correlation between them. In Fig. 2, it is highlighted the PCA configuration to cover 99% of the variance of the data, where the upper line represents the cumulative variance covered, while the lower one represents the variance covered per component. However, knowing the structure and the complexity of the data analysed, the PCA may not represent the most suitable data transformation to improve the learning models performances. In fact, by observing Fig. 2, one could argue that the PCA is not producing the desired effects (i.e. reducing the number of features without losing too much information) since most of the principal components are not so informative.

In this context, and as demonstrated by the following evaluation Sect.3.3, blindly maximising the variance amongst the data leads to worse performances. This phenomena suggest

[3] The IG, is purely theoretic, it does not consider any particular classification algorithm.

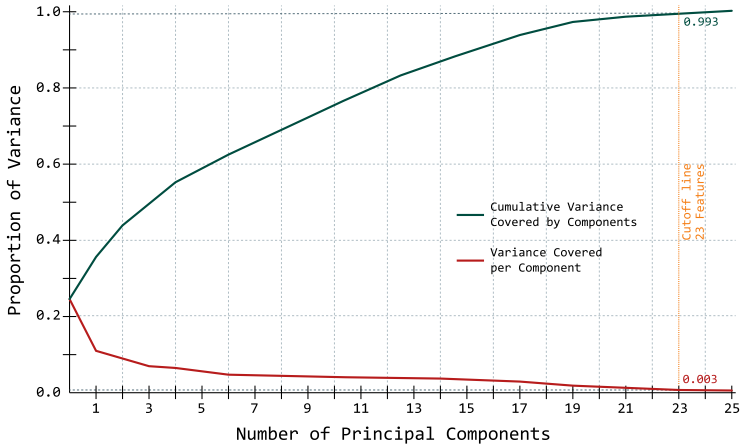
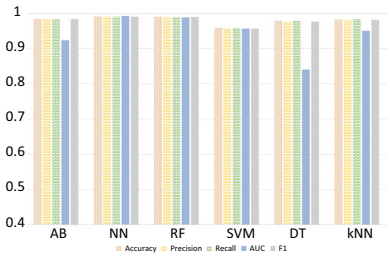
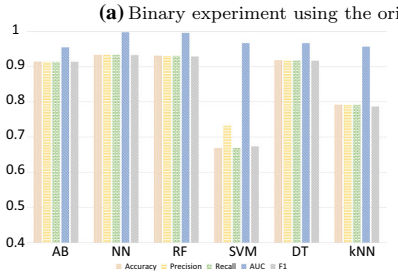


Fig. 2 PCA configured to cover 99% of the data variance



Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.983	0.983	0.983	0.923	0.983
NN	0.990	0.990	0.990	0.992	0.990
RF	0.989	0.989	0.989	0.988	0.989
SVM	0.958	0.955	0.958	0.956	0.956
DT	0.978	0.976	0.978	0.840	0.976
kNN	0.982	0.981	0.982	0.950	0.981

(a) Binary experiment using the original feature set. (Exp. 1, U.C. 1)



Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.913	0.913	0.913	0.954	0.913
NN	0.932	0.932	0.932	0.997	0.932
RF	0.930	0.928	0.930	0.995	0.928
SVM	0.668	0.733	0.668	0.966	0.673
DT	0.917	0.916	0.917	0.966	0.916
kNN	0.791	0.790	0.791	0.956	0.786

(b) Multiclass experiment using the original feature set. (Exp. 2, U.C. 1)

Fig. 3 Evaluation using the original feature set (U.C. 1)

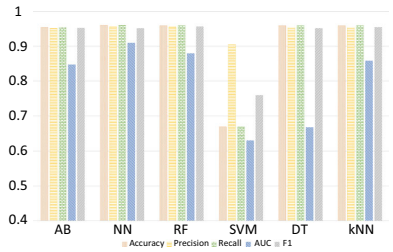
that an exploratory analysis regarding the nonlinear dimensionality reduction techniques may be indeed necessary.

3.3 Evaluation

In order to compare and confirm the findings of Sects. 2 and 3.2, we designed a series of experiments. These experiments were conducted on a Dell M3800 workstation with an Intel i7-4712HQ processor running at 2.30GHz, 16 GB of DDR3 RAM at 1600 MHz and a NVIDIA Quadro K1100M graphic processor. Experiments were run using Orange3 (Demšar et al. 2013), and for each set of data described in the aforementioned methodology Sect. 3.1, six classifiers were applied and cross-validated using a stratified tenfold approach. On the one hand, the six algorithms used are defined using the following configuration:

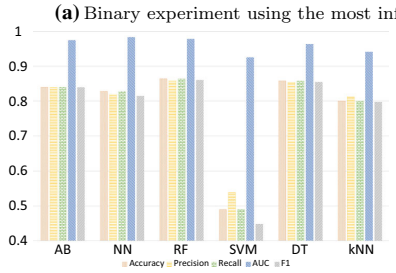
- *AdaBoost (AB)*—Using 50 trees as base estimators, with SAMME.R classificator (updates base estimators weight with probability estimates) and linear regression loss function.
- *Neural Network (NN)*—Single hidden layer with 100 nodes activated with the Rectified Linear unit (ReLu) function, weight optimised with the stochastic gradient-

Fig. 4 Evaluation using the most informative features (U.C. 2)



Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.954	0.951	0.954	0.847	0.952
NN	0.960	0.956	0.960	0.909	0.951
RF	0.959	0.955	0.959	0.879	0.956
SVM	0.669	0.905	0.669	0.630	0.759
DT	0.959	0.953	0.959	0.667	0.951
kNN	0.959	0.953	0.959	0.858	0.954

(a) Binary experiment using the most informative feature set. (Exp. 1, U.C. 2)



Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.840	0.839	0.840	0.974	0.839
NN	0.828	0.819	0.828	0.983	0.814
RF	0.864	0.859	0.864	0.978	0.860
SVM	0.490	0.538	0.490	0.925	0.448
DT	0.858	0.853	0.858	0.963	0.854
kNN	0.800	0.812	0.800	0.941	0.797

(b) Multiclass experiment using the most informative feature set. (Exp. 2, U.C. 2)

based optimiser (Adam), $\alpha = 0.0010$ and 200 max iterations.
- *Random Forest (RF)*—Using 10 trees, considering up to five attributes at each split and without splitting subsets smaller than five.
- *Support Vector Machine (SVM)*—Configured with $C = 1.00$, $\epsilon = 0.10$ and using the RBF Kernel.
- *Decision Tree (DT)*—Two minimum instances in leaves, do not split trees smaller than five, having a max depth of 100. Exiting condition when the majority reaches 95%.
- *k-Nearest Neighbours (kNN)*—Five neighbours using the Euclidean metric and a uniform weight.

While, on the other hand, the tables reported in Figs. 3, 4 and 5 are reporting the average values over the tested classes with a testing set obtained by cross-validation sampling (having 10 stratified folds).

Section 3.3.1 introduces the three evaluation use cases which are later discussed and commented in Sect. 3.3.2.

3.3.1 Experiments design and use cases

Two sets of experiments were run on with the same data set and the same configuration. In the first one, the malware families (reported in Sect. 3.1) are used as separate classes, while in the second one the malwares are considered as a single class, enabling the binary analysis of malware versus legitimate domain names. To be more specific:

Experiment 1 ***(Binary)*** *The binary experiment is designed to answer the ML question of separating legitimate FQDNs from malicious AGDs, considering all malware families as a single category.*

Experiment 2 ***(Multiclass)*** *The multiclass experiment is designed to go beyond the above-mentioned binary experiment in order to classify not only the legitimate FQDN but also sort malware samples according to their families.*

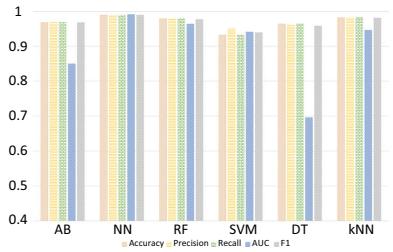
Moreover, in order to present the experiments results in combination with the aforementioned feature sets, three use cases are introduced and commented below.

Use Case 1 ***(Original features set)*** *In the first use case, both experiments were run using the original data set with the all the collected features, as presented in Table 8. The results are reported in Fig. 3. Specifically, it is possible to compare the results of the binary experiment and the multiclass one in Fig. 3a and b, respectively.*

Use Case 2 ***(Most informative features set)*** *Similarly, in the second use case the experiments were run using the data set updated with the features selected in the previous Sect. 3.2 and presented in Fig. 4, also divided into binary (Fig. 4a) and multiclass (Fig. 4b) experiments.*

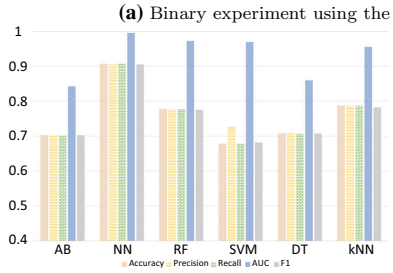
Use Case 3 ***(PCA features set)*** *Finally, and as reported above, the third use case features the experiments run using the data set manipulated with the PCA algorithm. Likewise, Fig. 5 presents the comparison tables and the relative charts for binary (Fig. 5a) and multiclass (Fig. 5b) experiments.*

Fig. 5 Evaluation using the PCA (U.C. 3)



Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.968	0.968	0.968	0.850	0.968
NN	0.989	0.989	0.989	0.991	0.989
RF	0.979	0.978	0.979	0.964	0.977
SVM	0.932	0.950	0.932	0.941	0.939
DT	0.964	0.961	0.964	0.696	0.958
kNN	0.982	0.981	0.982	0.946	0.981

(a) Binary experiment using the PCA feature set. (Exp. 1, U.C. 3)



Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.702	0.702	0.702	0.842	0.702
NN	0.906	0.905	0.906	0.995	0.905
RF	0.777	0.775	0.777	0.972	0.775
SVM	0.678	0.726	0.678	0.969	0.681
DT	0.707	0.708	0.707	0.859	0.707
kNN	0.787	0.787	0.787	0.955	0.782

(b) Multiclass experiment using the PCA feature set. (Exp. 2, U.C. 3)

3.3.2 Discussion

When looking at the multiclass experiments (Exp. 2), it is important to restate that the values reported in Figs. 3b, 4b and 5b are referring to the average value calculated over the classes-specific values. As a consequence, values such as the F1 score are not to be considered inconsistent; that is, the average of all the F1 scores over the classes might be outside the range delimited by the average of the precision and recall scores. The same applies to the AUC results, a value that represents the discrimination, i.e. the ability of the specific classification algorithm to sort the member of the target class. Not surprisingly, by having 17 classes (16 malware families plus the legitimate one, see Sect. 3.1) the average value of the AUC is quite high because of the excellent performances with regard to some easily identifiable malwares.

As expected, the multiclass experiments are presenting worse performances than the binary ones due to the poor performances obtained when distinguishing similar malwares such as Qakbot or Matsnu. In the binary experiments, however, those samples are considered as a generic malware class, thus improving the overall detection. Similarly, the performances of the PCA use case (U.C. 3) are worse than the ones reported in the original use case (U.C. 1). This worsening trend is mainly due to the nature of the PCA itself, i.e. maximise the variability of the features, but not the classes' separability. Although very common and widely used, the PCA is an example of unsupervised analysis tool that is not suitable in this scenario, thus leading in general to worse performances. Further researches are required in order to establish which feature reduction strategy optimally approximates the data. Preliminary results using nonlinear feature reduction techniques seem promising.

In contrast with the expected results of the previous use cases, the feature selection use case (U.C. 2) reports unanticipated slightly worse performances. However, the theory of ML permits to explain such results, especially when comparing them to the ones described in the literature (outlined in Sect. 2). First and foremost, the data set used plays an important role; that is, by accurately selecting the data involved in the analysis, it is possible to boost the accuracy and generally increase the performances. Secondly, different algorithms require different optimal structures for the data, e.g. RF are indifferent to scaling and normalisation techniques, while NN require normal and scaled distributions. Last, but not least, fine-tuning each algorithm permits to extend further their efficiency (Mantovani et al. 2015).

To sum up, this evaluation process once again pointed out the lack of common elements that can be used so as to guarantee the reproducibility of the experiments and thus their comparative analysis. To this extent, a publicly available data set of AGDs may lead to better, but more importantly, comparable results.

4 Conclusions and future work

Although DGA-based botnets have been proved challenging, there are some positive results within the research field

that looks promising. In this context, in fact, ML approaches have been successfully employed both in a supervised and unsupervised manner aiming to achieve usable and scalable solutions.

After presenting the state of the art in terms of both algorithms and feature set, this work has defined and established a common-ground knowledge regarding the features used as a base for such algorithms, showing that the Context-Free feature family is more than capable of pinpointing DGA-based malwares without harming the users' privacy. Besides what is discussed in Sect. 3.3.2, the sole use of Context-Free features achieves excellent results in terms of performances, precision and recall both in the Binary and in the Multiclass scenarios.

Therefore, the main outcomes of this work are several. In fact, this research firstly collects and studies the most relevant literature works that attempted to deal with the DGA-based botnet detection problem; secondly, this research item enables the development of privacy-preserving solutions[4]; and, finally, by collecting and studying the literature and the proposed features, it prearranges a common ground that enables future researches to focus on the evaluation and the creation of new efficient detection algorithms in the subject of DGA-based botnet. To this extent, in fact, preliminary analyses carried out internally indicate positive and interesting results.

Therefore, to summarise and complement what is proposed in Sect. 2.3, future works might include (i) the study of the Context-Aware feature family to establish whether it may combine and consolidate detection solutions; (ii) the development of a full-fledged, public available and labelled data set of either Context-Aware and Context-Free features that might emerge as a common ground for the evaluation of existing and new ML solutions; (iii) the exploratory analysis of the above-mentioned data with nonlinear techniques in order to achieve improved classification results; and, finally, (iv) testing detection algorithms against 0-day resilience by adding both new malware families and variants.

Acknowledgements This study was founded by a predoctoral and a postdoctoral INCIBE Grant within the "Ayudas para la Excelencia de los Equipos de Investigación Avanzada en Ciberseguridad" program, with Codes INCIBEI-2015-27353 and INCIBEI-2015-27352.

Compliance with ethical standards

Conflict of interest The authors declare that they do not have any conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

[4] By experimentally demonstrating that users' data are not strictly required to recognise malwares in the wild. See Sect. 3.3.

References

Abakumov A (2016) andrewaeva/DGA. URL https://github.com/andrewaeva/DGA

Abbink J, Doerr C (2017) Popularity-based detection of domain generation algorithms. In: 12th international conference on availability, reliability and security, pp 79:1–79:8. https://doi.org/10.1145/3098954.3107008

Abdel-Hamid O, Mohamed Ar, Jiang H, Deng L, Penn G, Yu D (2014) Convolutional neural networks for speech recognition. IEEE/ACM Trans Audio Speech Lang Process 22(10):1533–1545. https://doi.org/10.1109/TASLP.2014.2339736

Ahluwalia A, Traore I, Ganame K, Agarwal N (2017) Detecting broad length algorithmically generated domains. In: Intelligent, secure, and dependable systems in distributed and cloud environments, chap. 2, pp 19–34. Springer International Publishing. https://doi.org/10.1007/978-3-319-69155-8_2

Alieyan K, ALmomani A, Manasrah A, Kadhum MM (2017) A survey of botnet detection based on DNS. Neural Comput Appl 28(7):1541–1558. https://doi.org/10.1007/s00521-015-2128-0

Almomani A, Alauthman M, Albalas F, Dorgham O, Obeidat A (2018) An online intrusion detection system to cloud computing based on Neucube algorithms. Int J Cloud Appl Comput 8(2):96–112. https://doi.org/10.4018/IJCAC.2018040105

Anderson HS, Woodbridge J, Filar B (2016) DeepDGA: adversarially-tuned domain generation and detection. In: 2016 ACM workshop on artificial intelligence and security, pp 13–21. https://doi.org/10.1145/2996758.2996767

Antonakakis M, Perdisci R, Nadji Y, Vasiloglou N, Abu-Nimeh S, Lee W, Dagon D (2012) From throw-away traffic to bots: detecting the rise of DGA-based malware. In: 21st USENIX security symposium, pp 491–506. Bellevue, WA. URL https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/antonakakis

Bader J. Domain Generation Algorithms. URL https://github.com/baderj/domain_generation_algorithms

Baruch M, David G (2018) Domain generation algorithm detection using machine learning methods. In: Cyber security: power and technology, pp 133–161. Springer International Publishing. https://doi.org/10.1007/978-3-319-75307-2_9

Berger A, Gansterer WN (2013) Modeling DNS agility with DNSMap. In: 2013 proceedings IEEE INFOCOM, pp 3153–3158. https://doi.org/10.1109/INFCOM.2013.6567130

Biglar Beigi E, Hadian Jazi H, Stakhanova N, Ghorbani AA (2014) Towards effective feature selection in machine learning-based botnet detection approaches. In: 2014 IEEE conference on communications and network security, pp 247–25. https://doi.org/10.1109/CNS.2014.6997492

Bilge L, Sen S, Balzarotti D, Kirda E, Kruegel C (2014) Exposure: a passive DNS analysis service to detect and report malicious domains. ACM Trans Inf Syst Secur 16(4):14:1–14:28. https://doi.org/10.1145/2584679

Bishop C (2006) Pattern recognition and machine learning. Springer, Berlin

Bisio F, Saeli S, Lombardo P, Bernardi D, Perotti A, Massa D (2017) Real-time behavioral DGA detection through machine learning. In: 2017 international carnahan conference on security technology, pp 1–6. https://doi.org/10.1109/CCST.2017.8167790

Bugiel S, Nürnberger S, Pöppelmann T, Sadeghi AR, Schneider T (2011) AmazonIA: when elasticity snaps back. In: 18th ACM conference on computer and communications security, pp 389–400. https://doi.org/10.1145/2046707.2046753

Demšar J, Curk T, Erjavec A, Gorup Č, Hočevar T, Milutinovič M, Možina M, Polajnar M, Toplak M, Starič A, Štajdohar M, Umek L, Žagar L, Žbontar J, Žitnik M, Zupan B (2013) Orange: data

mining toolbox in Python. J Mach Learn Res 14:2349–2353. URL http://jmlr.org/papers/v14/demsar13a.html
Fran E, Hall MA, Witten IH (2016) The WEKA Workbench. Tech. rep. URL https://www.cs.waikato.ac.nz/ml/weka
Fu Y, Yu L, Hambolu O, Ozcelik I, Husain B, Sun J, Sapra K, Du D, Beasley CT, Brooks RR (2017) Stealthy domain generation algorithms. IEEE Trans Inf Forensics Secur 12(6):1430–1443. https://doi.org/10.1109/TIFS.2017.2668361
García S, Grill M, Stiborek J, Zunino A (2014) An empirical comparison of botnet detection methods. Comput Secur 45:100–123. https://doi.org/10.1016/j.cose.2014.05.011
Grill M, Nikolaev I, Valeros V, Rehak M (2015) Detecting DGA malware using NetFlow. In: 2015 IFIP/IEEE international symposium on integrated network management, pp 1304–1309. https://doi.org/10.1109/INM.2015.7140486
Gupta B, Agrawal DP, Yamaguchi S (eds) (2016) Handbook of research on modern cryptographic solutions for computer and cyber security, 1st edn. IGI Global
Han C, Zhang Y (2017) CODDULM: an approach for detecting C&C domains of DGA on passive DNS traffic. In: 2017 6th international conference on computer science and network technology, pp 385–388. https://doi.org/10.1109/ICCSNT.2017.8343724
Haykin S (1998) Neural networks: a comprehensive foundation, 2nd edn. Prentice Hall PTR, Upper Saddle River
Holz T, Steiner M, Dahl F, Biersack E, Freiling F (2008) Measurements and mitigation of peer-to-peer-based Botnets: a case study on storm worm. In: USENIX security 2008. URL https://www.usenix.org/conference/leet-08/measurements-and-mitigation-peer-peer-based-botnets-case-study-storm-worm
Hussain SA, Fatima M, Saeed A, Raza I, Shahzad RK (2017) Multilevel classification of security concerns in cloud computing. Appl Comput Inform 13(1):57–65. https://doi.org/10.1016/j.aci.2016.03.001
Kintis P, Miramirkhani N, Lever C, Chen Y, Romero-Gómez R, Pitropakis N, Nikiforakis N, Antonakakis M (2017) Hiding in plain sight: a longitudinal study of combosquatting abuse. In: ACM SIGSAC conference on computer and communications security, pp 569–586. https://doi.org/10.1145/3133956.3134002
Kührer M, Rossow C, Holz T (2014) Paint it black: evaluating the effectiveness of malware blacklists. In: RAID 2014: research in attacks, intrusions and defenses, June, pp 1–21. Springer International Publishing. https://doi.org/10.1007/978-3-319-11379-1_1
Leelasankar K, Chellappan C, Sivasankar P (2018) Handbook of research on network forensics and analysis techniques, chap. successful computer forensics analysis on the cyber attack Botnet, pp 266–281. IGI Global. https://doi.org/10.4018/978-1-5225-4100-4.ch014
Lerner Z (2014) Microsoft the Botnet hunter: the role of public-private partnerships in mitigating Botnets. Harvard J Law Technol 28(1):237–261. URL http://jolt.law.harvard.edu/articles/pdf/v28/28HarvJLTech237.pdf
Lobato AGP, Lopez MA, Sanz IJ, Cardenas AA, Duarte OCMB, Pujolle G (2018) An Adaptive real-time architecture for zero-day threat detection. In: 2018 IEEE international conference on communications (ICC), pp 1–6. https://doi.org/10.1109/ICC.2018.8422622
Luo X, Wang L, Xu Z, Yang J, Sun M, Wang J (2017) DGASensor: fast detection for DGA-based malwares. In: 5th international conference on communications and broadband networking, pp 47–53. https://doi.org/10.1145/3057109.3057112
Mac H, Tran D, Tong V, Nguyen LG, Tran HA (2017) DGA Botnet detection using supervised learning methods. In: 8th international symposium on information and communication technology, pp 211–218. https://doi.org/10.1145/3155133.3155166
Majestic-12 Ltd: The Majestic Million (2018) URL https://majestic.com/reports/majestic-million
Malware Domain List (2009) URL https://www.malwaredomainlist.com/mdl.php
Mantovani RG, Rossi AL, Vanschoren J, Bischl B, Carvalho AC (2015) To tune or not to tune: recommending when to adjust SVM hyper-parameters via meta-learning. In: Proceedings of the international joint conference on neural networks, vol 2015-September, pp 1–8. https://doi.org/10.1109/IJCNN.2015.7280644
Mell P, Grance T (2011) The NIST definition of cloud computing, NIST Special Publication 800-145. URL http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf
Mowbray M, Hagen J (2014) Finding domain-generation algorithms by looking at length distribution. In: 2014 IEEE international symposium on software reliability engineering workshops, pp 395–400. https://doi.org/10.1109/ISSREW.2014.20
Nespoli P, Papamartzivanos D, Mrmol FG, Kambourakis G (2018) Optimal countermeasures selection against cyber attacks: a comprehensive survey on reaction frameworks. IEEE Commun Surv Tutor 20(2):1361–1396. https://doi.org/10.1109/COMST.2017.2781126
Netlab 360: DGA Families. URL http://data.netlab.360.com/dga/
Nguyen TD, Cao TD, Nguyen LG (2015) DGA Botnet detection using collaborative filtering and density-based clustering. In: 6th international symposium on information and communication technology, pp 203–209. https://doi.org/10.1145/2833258.2833310
OSINT: OSINT DGA List. URL http://osint.bambenekconsulting.com/feeds/
Pelleg D, Moore A (2000) X-means: Extending K-Means with efficient estimation of the number of clusters. In: 7th international conference on machine learning pp 727–734. https://doi.org/10.1007/3-540-44491-2_3
Plohmann D (2015) DGArchive. URL https://dgarchive.caad.fkie.fraunhofer.de
Plohmann D, Yakdan K, Klatt M, Bader J, Gerhards-Padilla E (2016) A comprehensive measurement study of domain generating malware. In: 25th USENIX security symposium, pp 263–278. Austin, TX. URL https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_plohmann.pdf
Pu Y, Chen X, Pu Y, Shi J (2015) A clustering approach for detecting auto-generated Botnet domains. In: Applications and techniques in information security, pp 269–279. https://doi.org/10.1007/978-3-662-48683-2_24
Risk Analytics: DNS-BH-Malware Domain Blocklist (2007). URL http://www.malwaredomains.com
Schales DL, Jang J, Wang T, Hu X, Kirat D, Wuest B, Stoecklin MP (2016) Scalable analytics to detect DNS misuse for establishing stealthy communication channels. IBM J Res Dev 60(4):3:1–3:14. https://doi.org/10.1147/JRD.2016.2557639
Schiavoni S, Maggi F, Cavallaro L, Zanero S (2014) Phoenix: DGA-based Botnet tracking and intelligence. In: 11th international conference on detection of intrusions and malware, and vulnerability assessment, pp 192–211. Springer International Publishing. https://doi.org/10.1007/978-3-319-08509-8_11
Sharieh A, Albdour L (2017) A heuristic approach for service allocation in cloud computing. Int J Cloud Appl Comput 7(4):60–74. https://doi.org/10.4018/IJCAC.2017100104
Shi Y, Chen G, Li J (2017) Malicious domain name detection based on extreme machine learning. Neural Process Lett. https://doi.org/10.1007/s11063-017-9666-7
Song WJ, Li B (2016) A method to detect machine generated domain names based on random forest algorithm. In: 2016 international conference on information system and artificial intelligence, pp 509–513. https://doi.org/10.1109/ISAI.2016.0114
Stergiou C, Psannis KE, Kim BG, Gupta B (2018) Secure integration of IoT and cloud computing. Future Gener Comput Syst 78(3):964–975. https://doi.org/10.1016/j.future.2016.11.031

Stevanovic M, Pedersen JM, D'Alconzo A, Ruehrup S, Berger A (2015) On the ground truth problem of malicious DNS traffic analysis. Comput Secur 55:142–158. https://doi.org/10.1016/j.cose.2015.09.004

Stevanovic M, Pedersen JM, D'Alconzo A, Ruehrup S (2017) A method for identifying compromised clients based on DNS traffic analysis. Int J Inf Secur 16(2):115–132. https://doi.org/10.1007/s10207-016-0331-3

Thomas M, Mohaisen A (2014) Kindred domains: detecting and clustering Botnet domains using DNS traffic. In: 23rd international conference on World Wide Web, pp 707–712. https://doi.org/10.1145/2567948.2579359

Tong V, Nguyen G (2016) A method for detecting DGA Botnet based on semantic and cluster analysis. In: 7th symposium on information and communication technology, pp 272–277. https://doi.org/10.1145/3011077.3011112

Tran D, Mac H, Tong V, Tran HA, Nguyen LG (2018) A LSTM based framework for handling multiclass imbalance in DGA Botnet detection. Neurocomputing 275:2401–2413. https://doi.org/10.1016/j.neucom.2017.11.018

Truong D, Cheng G (2016) Detecting domain-flux botnet based on DNS traffic features in managed network. Secur Commun Netw 9(14):2338–2347. https://doi.org/10.1002/sec.1495

Tu TD, Guang C, Xin LY (2015) Detecting Bot-infected machines based on analyzing the similar periodic DNS queries. In: 2015 international conference on communications, management and telecommunications, pp 35–40. https://doi.org/10.1109/ComManTel.2015.7394256

Vinayakumar R, Soman K, Poornachandran P, Sachin Kumar S (2018) Evaluating deep learning approaches to characterize and classify the DGAs at scale. J Intell Fuzzy Syst 34(3):1265–1276. https://doi.org/10.3233/JIFS-169423

Vormayr G, Zseby T, Fabini J (2017) Botnet communication patterns. IEEE Commun Surv Tutor 19(4):2768–2796. https://doi.org/10.1109/COMST.2017.2749442

Watkins L, Beck S, Zook J, Buczak A, Chavis J, Robinson WH, Morales JA, Mishra S (2017) Using semi-supervised machine learning to address the big data problem in DNS networks. In: 2017 IEEE 7th annual computing and communication workshop and conference, pp 1–6. https://doi.org/10.1109/CCWC.2017.7868376

Woodbridge J, Anderson HS, Ahuja A, Grant D (2016) Predicting domain generation algorithms with long short-term memory networks. CoRR **abs/1611.0**. URL http://arxiv.org/abs/1611.00791

Xu S, Li S, Meng K, Wu L, Ding M (2017) An adaptive malicious domain detection mechanism with DNS traffic. In: 2017 VI international conference on network, communication and computing, pp 86–91. https://doi.org/10.1145/3171592.3171595

Yadav S, Reddy AKK, Reddy ALN, Ranjan S (2010) Detecting algorithmically generated malicious domain names. In: 10th ACM SIGCOMM conference on internet measurement, pp 48–61. https://doi.org/10.1145/1879141.1879148

Zhang S, Zhang X, Ou X (2014) After we knew it: empirical study and modeling of cost-effectiveness of exploiting prevalent known vulnerabilities across IaaS cloud. In: 9th ACM symposium on information, computer and communications security, pp 317–328. https://doi.org/10.1145/2590296.2590300

Zhang H, Gharaibeh M, Thanasoulas S, Papadopoulos C (2016) BotDigger: detecting DGA Bots in a single network. Tech. rep., Colorado State University. URL http://www.cs.colostate.edu/TechReports/Reports/2016/tr16-101.pdf

2

UMUDGA: A dataset for profiling DGA-based botnet

Title:	UMUDGA: A dataset for profiling DGA-based botnet
Authors:	Mattia Zago, Manuel Gil Pérez, Gregorio Martínez Pérez
Journal:	Computers & Security
J.I.F.:	3.579 Q2 (2019)
Publisher:	Elsevier
Volume:	92
Pages:	101719
Year:	2020
Month:	May
DOI:	10.1016/j.cose.2020.101719
Status:	Published

Abstract

Advanced botnet threats are natively deploying concealing techniques to prevent detection and sinkholing. To tackle them, machine learning solutions have become a standard approach, especially when dealing with Algorithmically Generated Domain (AGD) names. Nevertheless, machine learning state-of-the-art is non-specialist at best, having multiple issues in terms of rigorousness, reproducibility and ultimately credibility. This research focuses on the first critical step of the training phase, that is, the collection of data suitable for being analysed by algorithms. We have detected a common lack of scientific rigorousness in the literature regarding the aforementioned AGD analysis and, therefore, we advocate two major contributions in this article: *i)* a thorough analysis of the cyber panorama in terms of botnets that make use of Domain Generation Algorithms (DGAs) as evasive techniques, that flows into *ii)* a full-fledged machine-learning-ready labelled dataset that features over 30 million AGDs sorted in 50 malware variant classes. This mature dataset aims to fill the gap in the comparability between the different researches published in the literature. Lastly, two minor contributions are also included in this article: *iii)* we designed an exploratory analysis of the proposed dataset to provide both data characteristics and potential future research lines, which eventually emerges as *iv)* a collection of suggested guidelines. When proposing a machine learning solution, researchers should adhere to it in order to achieve scientific rigorousness.

Keywords

Domain Generation Algorithm (DGA) · Natural Language Processing (NLP) · Machine Learning · Botnet · Network Security

Computers & Security 92 (2020) 101719

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

UMUDGA: A dataset for profiling DGA-based botnet

Mattia Zago, M.Sc.*, Manuel Gil Pérez, Ph.D., Gregorio Martínez Pérez, Ph.D.

Department of Information Engineering and Communications, University of Murcia, Campus de Espinardo s/n, 30100 Murcia, Spain

ARTICLE INFO

Article history:
Received 16 August 2019
Revised 5 November 2019
Accepted 16 January 2020
Available online 18 January 2020

Keywords:
Domain Generation Algorithm (DGA)
Natural Language Processing (NLP)
Machine learning
Botnet
Network security

ABSTRACT

Advanced botnet threats are natively deploying concealing techniques to prevent detection and sinkholing. To tackle them, machine learning solutions have become a standard approach, especially when dealing with Algorithmically Generated Domain (AGD) names. Nevertheless, machine learning state-of-the-art is non-specialist at best, having multiple issues in terms of rigorousness, reproducibility and ultimately credibility. This research focuses on the first critical step of the training phase, that is, the collection of data suitable for being analysed by algorithms. We have detected a common lack of scientific rigorousness in the literature regarding the aforementioned AGD analysis and, therefore, we advocate two major contributions in this article: *i)* a thorough analysis of the cyber panorama in terms of botnets that make use of Domain Generation Algorithms (DGAs) as evasive techniques, that flows into *ii)* a full-fledged machine-learning-ready labelled dataset that features over 30 million AGDs sorted in 50 malware variant classes. This mature dataset aims to fill the gap in the comparability between the different researches published in the literature. Lastly, two minor contributions are also included in this article: *iii)* we designed an exploratory analysis of the proposed dataset to provide both data characteristics and potential future research lines, which eventually emerges as *iv)* a collection of suggested guidelines. When proposing a machine learning solution, researchers should adhere to it in order to achieve scientific rigorousness.

1. Introduction

The 2019 cybersecurity landscape is seriously perilous, in fact, as for the past few years, several technical reports from major security stakeholders (Brandt et al., 2018; Eremin, 2018; Fireeye Mandiant Services, 2019; Kujawa et al., 2019; O'Gorman et al., 2019; Spamhaus Project, 2019a; 2019b) have confirmed that cybercriminal activities are rising in almost every sector. Although 2018 has seen a decrease in the number of malware variants available in exploit kits (-63% according to O'Gorman et al. (2019)), the threat posed by botnets is increasing (Kujawa et al., 2019; Spamhaus Project, 2019a; 2019b), especially when considering the ones with backdoor functionalities (+34% in private *vs* +173% in enterprises (Kujawa et al., 2019)). One could argue that the reason behind such "positive result" (from the cybercriminals standpoint) is due to the extensive usage of well-known evasion techniques such as obfuscation, live encryption and Domain Generation Algorithm (DGA) (Etaher et al., 2015; Vormayr et al., 2017; Zago et al., 2019a). These techniques are employed by almost every major malware in the wild in order to bypass Intrusion Detection System (IDS) inspection measures since the infamous Kraken and Conficker malwares, back in 2008. To be more precise, a DGA is a technique that makes use of pseudo-random routines and external factors (such as time, data feeds, etc. Vormayr et al. (2017)) to generate multiple Fully Qualified Domain Names (FQDNs) to use as *rendezvous-point* (*i.e.*, algorithmically generated domains, or *algorithmically generated domains (AGDs)* for the botnets' Command & Control (CC) servers (Zago et al., 2019a). DGAs are a notably effective evasion technique that consists of generating thousands, often millions, of pseudo-random domain names. Their strength relies in the asymmetry of resource required by the attacker(s) (*i.e.*, the botmaster(s), the malicious actor(s) in control of the CC servers and thus in control of the botnet) and the defenders (*i.e.*, Internet Service Provider (ISP), cyber security vendors and, in general, the scientific community). That is to say, the defenders needs to detect and react against all AGDs, while the attacker(s) to be able to communicate with the botnet only require a single, undetected and working domain name.

To grasp the threat it is imperative to firstly understand its magnitude. That is to say, the most recent technical reports estimate the number of malicious FQDNs to be around 9.9% of the total domain names, of which, 1 in 5 belongs to DGA-based botnets (around 1.8% of all the registered domain names) (O'Gorman et al., 2019). To be more precise, according to the Spamhaus Project (2019b) (and confirmed by

* Corresponding author.
E-mail addresses: mattia.zago@um.es (M. Zago), mgilperez@um.es (M. Gil Pérez), gregorio@um.es (G. Martínez Pérez).

https://doi.org/10.1016/j.cose.2020.101719
0167-4048/

Plohmann et al. (2016), among others), the most abused domain names extensions are by far `.com` and `.uk`. Secondly, it is clear that the botnet specialisation are versatile and they normally change depending on the main actors behind them (Eremin, 2018). That is to say, botnets belonging to the same family are instantiated and managed by different operators, with varying objectives. In other words, botnets are offered as cloud-based malware-as-a-service (Putman et al., 2018). Although this is a well-known security issue (Hammi et al., 2019), accordingly to (Spamhaus Project, 2019a), ISPs are not following the best practices for customer verification enabling cybercriminals to automatic sign-up fraudulent accounts (61% of the observed CC servers in 2018). For example, Cloudflare has been identified as the most abused ISP for hosting CC servers (Spamhaus Project, 2019a).

Finally, and as previously mentioned, evasive techniques are widely used by botnets, especially DGA-based ones, to avoid detection. It has been estimated that the average dwell time, *i.e.*, the number of days an attacker is present on a victim network from first evidence of compromise to detection, is to be measured in months (Eremin, 2018; Fireeye Mandiant Services, 2019). Furthermore, the fact that most botnets present CC servers in multiple countries (njRAT, DarkComet and NanoCore malwares have them in over 80 countries) further demonstrate the limitation of sinkholing and, in general, reaction techniques.

In general, endpoint countermeasures (such as blacklisting) have already been proved ineffective (Kührer et al., 2014). Therefore, the cybersecurity community is actively researching and designing machine learning (ML)-based solutions to overcome this limitation. To be more precise, there are two potential areas of application of ML-powered products, namely the detection of actively queried AGDs and the dynamic reaction against either the malware spreading and/or the infected machine communications. This research focuses on this first area of application, that is the detection of AGDs in the wild using ML techniques. Our claim is that it is mandatory to shift the detection of this peculiar class of botnets from the attack phase (*i.e.*, when the infected machines are actively engaging in malicious activities) to the early stages, in which the botnets and the CC servers are being instantiated and configured.

The research inhere proposed leverages this idea, eventually aiming to enable cybersecurity operators to perform preemptive analysis of services to flag suspicious associated domain names. However, the very first step required in order to deploy a ML-powered solution to achieve this objective is to obtain trustworthy and reliable data to be used as a training set. As we sill demonstrate in the related work Section 2, this still represents a major challenge. In fact, as reported by Zago et al. (2019a) among others, the shortcoming of mature, ML-ready and publicly available datasets dedicated to DGA-based detection represent nowadays a critical setback.

Our proposal is to ease and eventually standardise the future researches on this subject by providing firstly a mature dataset (which will be publicly released as publicly available dataset (Zago et al., 2020), code repository (Zago et al., 2020) and documentation (Zago et al., 2019b)); secondly, a complete state-of-the-art in terms of potential third-party data source (Section 2); and, thirdly, an exploratory analysis to guide future practitioner toward the open challenges in terms of machine learning applications, which also have been previously discussed by the authors (Zago et al., 2019a).

To this extent, our endeavour is to propose the research community, but also the registrars, the ISPs and the cybersecurity vendors, to focus on creating innovative ML solutions for identifying DGA-based botnet threats.

Therefore, the main contributions of this research are twofold:

i) firstly, the in-depth state-of-the-art analysis regarding publicly available datasets that might provide the solid ground for ML-powered detection frameworks; and,
ii) secondly, the public release of a full-fledged, privacy-aware and ML-ready dataset, called `UMUDGA`, featuring samples from 50 malware variants for a total of 30+ million domain names.

Additionally, to support the findings and further extend the state-of-the-art, this article includes another two minor contributions, which may prove useful to future researchers, namely

iii) the exploratory analysis that serves both as data descriptor and as data breakdown and interpretation; and, finally,
iv) the discussion sprang from the analysis that flows into a collection of suggested guidelines that should be followed to achieve the required scientific rigorousness.

The structure of this article is the following. Section 2 introduces the comparison metrics and the current state-of-the-art in terms of publicly available datasets, while Section 3 presents the architecture, the methodology and in general, the characteristics of the proposed `UMUDGA` dataset. In addition, Section 4 reports a brief analysis of the data, and as a consequence, Section 4.3 pinpoints the main challenges identified and the guideline for future works. At last, Section 5 summarises-up and concludes the article.

2. Related works

Building a formal and strict comparison of the existent related works in terms of malware datasets that includes DGA-based botnets is a non-trivial research task. Moreover, the innate scope differences between them further aggravate the shortcoming of shared and acknowledged comparison techniques.

Nevertheless, in the past decade there are several notable researches that achieved to provide great support to the cybersecurity community that study innovative solutions for tackling network threats. The scope of this survey is to analyse them as ML data sources. As will become clear in the following sections, nearly half of the existing datasets are ready "out-of-the-box" to be used in ML-powered solutions. Besides the clear issues in terms of generality and representativeness, important weaknesses will be highlighted in terms of stating whenever a data source is verifiable, reproducible and extensible. To be more precise, we have collected 9 fundamental characteristics that a ML dataset should achieve according to both our view and the researches available in literature. In the following paragraphs, we will discuss these characteristics and highlight how the publicly available datasets struggle to excel in all of them. The outlines of such comparison can be found in Table 1. In the table, the last line is dedicated to summarise the achievements of our proposed dataset, `UMUDGA`, which properties will be discussed in Section 3.

Finally, a review of literature approaches in terms of ML solutions for tackling DGA-based botnets is available at Zago et al. (2019a).

The first and foremost is the property that establishes whether the dataset is composed of real data or it is an artificial artefact. We thus introduce the first property 2.1.

Definition 2.1 (Synthetic. `SYNT`) The dataset is artificially created either by generating the samples or by mixing multiple sources.

Firstly, the `SYNT` does not represent a binary "good *versus* bad" feature. Instead, both values are important and legitimate required depending of the application scope and purpose. That is to say that on the one hand, a ✗ value represents a dataset that makes use of data that has been organically captured from real networks and real infected machines (Bhuyan et al., 2015; Kent, 2015a, 2015b;

Table 1
Comparison of datasets embodying DGA-based botnets.

Name	Year	Characteristics								
		SYNT (2.1)	GNRL (2.2)	RPST (2.3)	BLNC (2.4)	EXTS (2.5)	VRFB (2.6)	PROR (2.7)	MLRD (2.8)	LABL (2.9)
PCAP Based										
ISCX IDS (Shiravi et al., 2012)	2012	✓	✗	≈	≈	✓	✓	✓	✗	✓
CTU (García et al., 2014)	2014	≈	✓	✗	✗	≈	✗	✓	≈	≈
CONTAGIO Parkour, 2015	2015	✗	✗	✗	✗	✗	✗	✗	✗	✓
CyberData1 Kent, 2015a, 2015b	2015	✗	≈	✓	✗	✗	✗	✓	✗	✗
ISOT HTTP Alenazi et al. (2017)	2017	✓	✗	≈	✗	≈	≈	≈	✗	✓
CTU Extended García et al. (2014)	2018	≈	✓	✗	✗	≈	✗	✓	≈	≈
Network Flows Based										
ISOT Zhao et al. (2013)	2013	✓	✗	≈	✗	≈	≈	≈	✓	✓
UNB Botnet Beigi et al. (2014)	2014	✓	✗	✗	✗	≈	≈	✓	✓	✓
PUF Sharma et al. (2018)	2018	✗	✗	≈	✗	✗	✗	✓	✓	✓
FQDN Based										
SuperCowPowers Wylie (2013)	2013	✓	✗	≈	≈	✓	✓	✓	≈	≈
Andrewaeva Abakumov (2014)	2014	✓	✗	✗	✗	✗	✗	✓	≈	✓
Pchaigno Chaignon (2015)	2015	✓	✗	≈	≈	✓	✓	✓	≈	✓
BaderJ Plohmann et al. (2016), Bader, 2019	2015	✓	✓	≈	≈	✓	✓	✓	≈	✓
AmritaDGA Vinayakumar et al. (2018)	2018	✓	≈	✓	✓	≈	✗	✓	≈	≈
Features Based										
NSL-KDD Tavallaee et al. (2009)	2009	✓	✗	✗	✓	✗	✗	✓	✓	✓
UCSD URL Ma et al. (2009)	2009	✗	≈	≈	✗	✗	✗	✓	✓	≈
UMUDGA Zago et al., 2020	2020	✓	✓	✓	✓	✓	✓	✓	✓	✓

Legend: ✓ Yes – ✗ No – ≈ Partially Characteristics abbreviations, as defined in Section 2: Synthetic (SYNT), General (GNRL), Representative (RPST), Balanced (BLNC), Extensible (EXTS), Verifiable (VRFB), Privacy-Orientation (PROR), Machine L6earning Ready (MLRD), Labelled (LABL).

Moustafa, 2017; Parkour, 2015); on the other hand, a ✓ value represents a synthetic dataset, that can be generated (with different degrees of randomness). The former requires to complete the considerable task of having the data labelled for training purposes, while the latter, if accurately and unbiased generated, might potentially represent a more reliable source of new labelled data.

It is important to notice that synthetic does not implies unrealistic data, however, it is possible that the generation process introduces dependencies that ML algorithms can detect (Beigi et al., 2014). That is to say that the action of mixing two different data sources does not imply better performances. To provide some examples, the following scenarios are considered synthetic:

i) the injection of previously captured malware traces in any traffic data; and
ii) the generation of malware data by executing the malware in a controlled environment.

As previously mentioned, and as summarised in Table 1, the dataset strictly composed by real data collected from real network are rare (CONTAGIO (Parkour, 2015), CyberData1 (Kent, 2015a, 2015b), PUF (Sharma et al., 2018) and UCSD URL (Ma et al., 2009)). Most authors prefer to inject malware data into background traffic, generate it in controlled environment, or mixing the captures from different sources (ISCX IDS (Shiravi et al., 2012), CTU (García et al., 2014), ISOT HTTP (Alenazi et al., 2017), ISOT (Zhao et al., 2013), UNB Botnet (Beigi et al., 2014) and NSL-KDD (Tavallaee et al., 2009)). A noticeable trend is identifiable with respect to the FQDN-based datasets, in which the data consists of AGD lists collected from multiple sources like security vendors or bulletins, often including also an implementation of the DGA (SuperCowPowers (Wylie, 2013), Andrewaeva (Abakumov, 2014), Pchaigno (Chaignon, 2015), BaderJ (Plohmann et al., 2016; Bader, 2019 and AmritaDGA (Vinayakumar et al., 2018)).

Secondly, in parallel with Property 2.1, there are the *General GNRL*, *Representative RPST* and *Balanced BLNC* characteristics, that reflect the realism of the dataset. Although a dataset cannot be at the same time synthetic and real, it surely can be both synthetic and realistic. In fact, by including a wide range of malware families (Property 2.2, GNRL, GNRL) (Beigi et al., 2014; Bhuyan et al., 2015; García et al., 2014; Moustafa, 2017; Shiravi et al., 2012), each one represented by a sizeable (Property 2.3, RPST) (Beigi et al., 2014; Bhuyan et al., 2015; Moustafa, 2017; Sharma et al., 2018; Shiravi et al., 2012) amount of samples, comparable to the other classes (Property 2.4, BLNC) (Berman et al., 2019), might result in a realistic representation of a real environment. Thus, the following definitions hold:

Definition 2.2 (General. GNRL) The dataset covers a wide range of malware families rather than being composed by a few specific examples. To be more precise, the volume of the data is enough to accurately represent a real-world scenario.

Definition 2.3 (Representative. RPST) The dataset includes, for every category, enough instances to accurately reflect the characteristics of the larger population.

Definition 2.4 (Balanced. BLNC) The dataset has a comparable number of samples for each category, *i.e.*, the number of instances belonging to a class should not outnumber any other class.

One could say that a dataset including several instances of a specific malware execution is representative of the variant (Property 2.3, RPST), but not general (Property 2.2, GNRL) nor balanced (Property 2.4, BLNC). Moreover, a dataset that features multiple instances of several malware variants might be general (Property 2.2, GNRL) with respect of the network threats, representative of the malware families examined (Property 2.3, RPST), but not balanced with respect to legitimate or background traffic (Property 2.4, BLNC). As previously mentioned, these three characteristics are quite difficult to find altogether in a dataset, in fact, most of the PCAP-based repositories are not general (Property 2.2, GNRL) nor representative (Property 2.3, RPST), let alone bal-

anced (Property 2.4, BLNC). Nevertheless, a few exceptions shine, even if only in a single category, namely the completeness of CTU (García et al., 2014) and BaderJ (Plohmann et al., 2016; Bader, 2019, in terms of number of malware families (Property 2.2, GNRL) and the amount of samples in each class represented by CyberData1 (Kent, 2015a, 2015b) and AmritaDGA (Vinayakumar et al., 2018) (Property 2.3, RPST). To the best of our knowledge, there is no dataset (apart from AmritaDGA (Vinayakumar et al., 2018) and NSL-KDD (Tavallaee et al., 2009)) that presents clearly balanced data samples.

Thirdly, to ensure the reusability and the consistency of the results derived from the dataset, two properties are defined aiming to:

i) measure whether the research community can extend and eventually enhance the data (Property 2.5, EXTS) (Bhuyan et al., 2015); and
ii) verify the data, when the whole replication process is not doable (Property 2.6, VRFB) (Bhuyan et al., 2015).

Definition 2.5 (Extensible. EXTS) The dataset is publicly available and well documented to enable the research community to extend or combine it with other data sources aiming to improve its reusability.

Definition 2.6 (Verifiable. VRFB) The data included in the dataset provide enough means to permit the research community to prove the consistency, the accuracy and the genuineness of the data, ideally resulting in a fully reproducible dataset.

Datasets composed by PCAP files have medium-to-low extensibility due to the fact that mixing it with other traffic sources might not result in a effective dataset. Moreover, dataset obtained by eavesdropping a real network is to be considered not verifiable and not replicable, because of the strong dependence from the context and the environmental conditions. However, a capture that is obtained from a controlled environment, *i.e.*, a testbed, is to be considered verifiable and potentially replicable, depending on the generation process. Finally, the simultaneous replay of multiple well-known traffic captures is to be considered both verifiable and replicable. Not surprisingly, both properties are generally satisfied by the FQDN-based category, while the other datasets generally achieve partial success at best. Notably, the ISCX IDS (Shiravi et al., 2012) is the only dataset in its category to achieve both properties.

Fourthly, as previously stated by (García et al., 2014; Zago et al., 2019a), a dataset should be made publicly available without doubts or concerns about harming users' privacy. Property 2.7, PROR, has been defined to target this aspect.

Definition 2.7 (Privacy-Orientation. PROR) The dataset has been designed to not include any privacy-harming content nor is required for the research community to harm the users' privacy in order to deploy or include the data in their experiments.

Any form of network traces that also includes the payload (*e.g.*, PCAP files) is natively including personal data that potentially harm the users' privacy. Although the network flows format dictates to strip the payload of the packets in order to aggregate them, the IP addresses are still considered to certain extent as personal information. As expected, almost all dataset composed by network flows, AGD lists or features do not contain personal information or user-specific data (UNB Botnet (Beigi et al., 2014), PUF (Sharma et al., 2018), SuperCowPowers (Wylie, 2013), Andrewaeva (Abakumov, 2014), Pchaigno (Chaignon, 2015), BaderJ (Plohmann et al., 2016; Bader, 2019, AmritaDGA (Vinayakumar et al., 2018), NSL-KDD (Tavallaee et al., 2009) and UCSD URL (Ma et al., 2009)). It is also interesting to note that, generally, the PCAP-based solutions presents some sort of data anonymisation even when they are not built as privacy-oriented solutions (ISCX IDS (Shiravi et al., 2012), CyberData1 (Kent, 2015a, 2015b), ISOT HTTP (Alenazi et al., 2017) and CTU (both original and Extended) (García et al., 2014)).

Finally, one of the scopes of this research is to explore the state-of-the-art oriented toward ML applications. To achieve this, two properties are defined to measure how easy it is to use "as-is" (Property 2.8, MLRD) (as suggested by Sharma et al. (2018)) and to indicate whether the dataset is labelled or not (Property 2.9, LABL) (Berman et al., 2019; Bhuyan et al., 2015; García et al., 2014; Moustafa, 2017; Sharma et al., 2018; Shiravi et al., 2012).

Definition 2.8 (Machine Learning Ready. MLRD) The dataset is composed by carefully curated samples. There are no missing values nor unwanted characters. Moreover, the data format is consistent across all the samples and it is suitable for usage with the leading tools.

Definition 2.9 (Labelled. LABL) Each sample is carefully characterised with one or more class attributes, eventually providing a variable granularity of the labels.

For example, a dataset composed by network flows is directly suitable of being directly plugged into ML solutions, provided that the target platform can support string and date features. By nature, both network flows based solutions (ISOT (Zhao et al., 2013), UNB Botnet (Beigi et al., 2014) and PUF (Sharma et al., 2018)) and feature based (NSL-KDD (Tavallaee et al., 2009) and UCSD URL (Ma et al., 2009)) ones are natively pluggable in ML algorithms, while FQDN lists (SuperCowPowers (Wylie, 2013), Andrewaeva (Abakumov, 2014), Pchaigno (Chaignon, 2015), BaderJ (Plohmann et al., 2016; Bader, 2019) and AmritaDGA (Vinayakumar et al., 2018)) are directly usable only by a subset of them (*e.g.*, deep learning or text processing). With regards to the labels, only CyberData1 (Kent, 2015a, 2015b) does not present any form of class separation. In order to provide an overall view, and as previously mentioned, in Table 1 it is possible to find the relevant datasets compared according to the aforementioned properties.

It is worth mentioning, that although not pinpointed in this section, our proposed dataset, UMUDGA, meets all the properties here described. A detailed discussion of such achievements is offered in Section 3.

Finally, for a few selected datasets (either for their importance or their properties) we realised also a quantitative analysis in terms of number of legitimate and malicious domain names and number of classes. Table 2 reports these findings. The validation column is obtained by processing each FQDN with Google Guava library, and specifically its `InternetDomainName` class's method which checks if the domain name is syntactically valid using lenient validation (The Guava Authors, 2009).

To be more precise, we excluded the remaining datasets for being obsolete (NSL-KDD (Tavallaee et al., 2009) and UCSD URL (Ma et al., 2009)), not labelled (CyberData1 (Kent, 2015a, 2015b)) or labels not aligned with the scope of this article (ISCX IDS (Shiravi et al., 2012) labels attacks, not malwares) and finally, for not being yet publicly released (PUF (Sharma et al., 2018)). With regards of the CTU (García et al., 2014) and the CTU Extended (García et al., 2014), we have considered them as a single dataset, extracting the data from both. Pchaigno (Chaignon, 2015) and BaderJ(Plohmann et al., 2016; Bader, 2019) are datasets of DGA, thus the quantitative comparison based on the number and properties of the FQDN is not applicable. To be more precise, and as reported both in Section 3 and (Zago et al., 2019b), the generators used in our dataset, UMUDGA, are using both (Chaignon, 2015) and Bader, 2019 as source, among others.

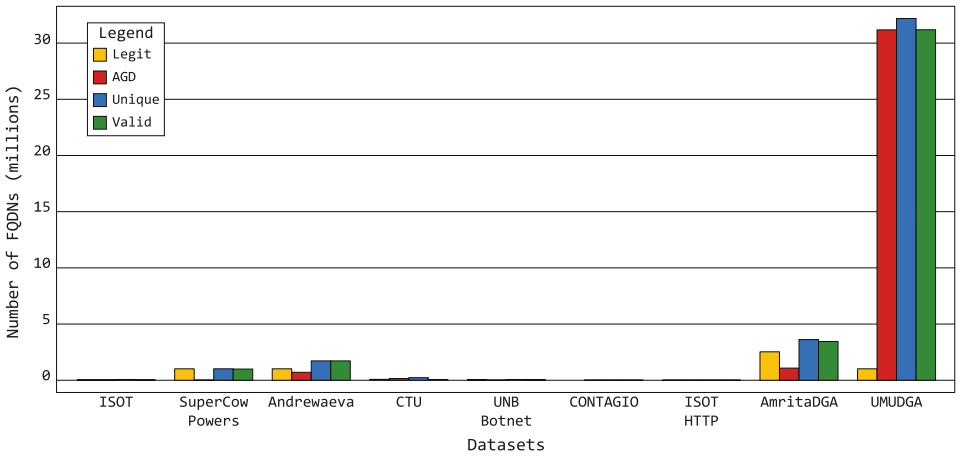
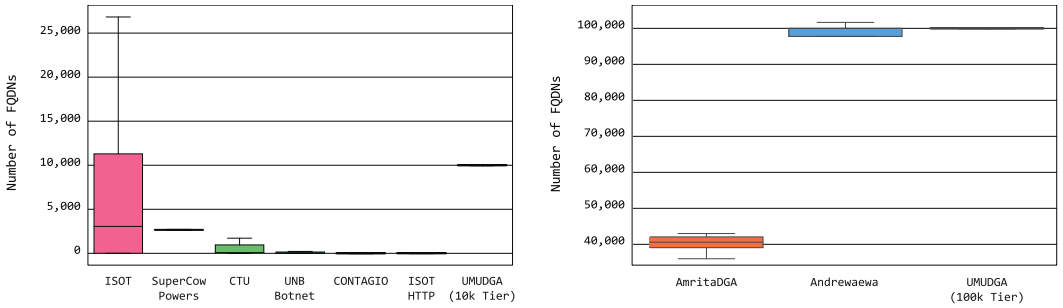


Fig. 1. Datasets metrics, as reported in Table 2.



(a) Datasets with less than 30 thousands FQDNs per class.

(b) Datasets with more than 30 thousands FQDNs per class.

Fig. 2. Boxplots representing the number of FQDN per class. The ideal dataset present a small interquartile range (*i.e.*, more balanced) and a high median (*i.e.*, more samples per class).

Table 2
Dataset comparison in terms of number of FQDN available for analysis and number of classes (including the legitimate one, when available).

Name	Year	Legit	AGD	Unique	Valid	Classes
ISOT (Zhao et al., 2013)	2013	30,699	32,952	63,651	31,297	5
SuperCowPowers (Wylie, 2013)	2013	1,000,000	2,670	1,002,670	986,081	2
Andrewaeva (Abakumov, 2014)	2014	1,000,000	694,173	1,694,173	1,694,167	9
CTU (García et al., 2014)	2014	73,020	153,999	227,019	62,620	8
UNB Botnet (Beigi et al., 2014)	2014	46,440	15,734	62,183	46,474	17
CONTAGIO (Parkour, 2015)	2015	0	8,612	3,637	3,620	110
ISOT HTTP (Alenazi et al., 2017)	2017	3,114	105	3,219	1,298	9
AmritaDGA (Vinayakumar et al., 2018)	2018	2,498,076	1,072,418	3,570,494	3,405,238	21
UMUDGA Zago et al., 2020	2020	1,000,000	30,799,449	31,799,449	30,799,449	51

The numbers reported in Table 2, have been graphically highlighted also in Fig. 1 (in terms of total number of samples) and in Fig. 2 (in terms of number of samples per class). In particular, Fig. 2a and b presents the different datasets with a suitable scale. In the figures, the ideal dataset presents a small interquartile range (all the classes have roughly the same amount of samples, *i.e.*, it satisfies Property 2.4, BLNC) and a high median (the average number of samples per class is elevated, thus potentially achieving Property 2.3, RPST). In both figures, outliers are not represented and our proposed UMUDGA has been included with the appropriate Tier (*i.e.*, the size of each class sample set, as will be described in Section 3).

Table 3
Comparison of datasets in terms of overlapping percentages.

Dataset	Year	Percentage of overlap with									
		Zhao et al. (2013)	Wylie (2013)	Abakumov (2014)	García et al. (2014)	Beigi et al. (2014)	Parkour, 2015	Alenazi et al. (2017)	Vinayakumar et al. (2018)	UMUDGA	Alexa
ISOT (Zhao et al., 2013)	2013	N.A.	12%	13%	16%	100%	1%	–	19%	10%	10%
SuperCowPowers (Wylie, 2013)	2013	–	N.A.	44%	1%	–	–	–	41%	22%	22%
Andrewaeva (Abakumov, 2014)	2014	–	25%	N.A.	–	–	–	–	38%	24%	18%
CTU (García et al., 2014)	2014	8%	10%	10%	N.A.	22%	1%	–	17%	8%	8%
UNB Botnet (Beigi et al., 2014)	2014	67%	9%	10%	29%	N.A.	2%	–	20%	7%	7%
CONTAGIO (Parkour, 2015)	2016	12%	9%	9%	21%	24%	N.A.	2%	34%	12%	7%
ISOT HTTP (Alenazi et al., 2017)	2017	3%	15%	16%	7%	12%	7%	N.A.	81%	16%	16%
AmritaDGA (Vinayakumar et al., 2018)	2018	–	12%	19%	–	–	–	–	N.A.	16%	12%
UMUDGA Zago et al., 2020	2020	–	1%	1%	–	–	–	–	2%	N.A.	3%

With Alexa we indicate the top one million domains Alexa Internet Inc, 2019, 2018 update. With "–" we indicate that the overlap is either non-existent or smaller than 0.1%.

In Table 2, the first two columns, namely the *legit* and the *AGD* columns, represent the amount of FQDN obtained from the lists or PCAP files, and thus do not include those databases that do not include domain names (Kent, 2015a, 2015b; Chaignon Bader, 2019; Ma et al., 2009; Tavallaee et al., 2009) and those that are not publicly available (Sharma et al., 2018). The *legit* column reports the number of FQDNs considered legitimate while the *AGD* one reports the number of malicious domains, using wireshark (Gerald Combs, 1998) to extract the `dns.qry.name` field for PCAP-based datasets. The two lists are then combined, sorted and all duplicates are removed, results are reported in the *unique* column. Moreover, the fourth column (*valid*) is obtained by processing each FQDN with Google Guava library, and specifically its `InternetDomainName` class's method which checks if the domain name is syntactically valid using lenient validation (The Guava Authors, 2009).

Finally, the datasets have been analysed according to their overlaps in terms of FQDNs and Table 3 identifies the amount of collisions registered within each dataset. To be more precise, the overlap is defined as the percentage of the dataset that is shared with the others, *i.e.*, giving any two lists of FQDNs, namely A and B, the percentage of collision is calculated as follows:

$$\text{overlap}(A, B) = 100 \cdot \frac{|A \cap B|}{|A|} \tag{1}$$

It is important to notice that the function is not symmetric, that is to say, a permutation of the input variables changes the result value, *i.e.*, overlap $(A, B) \neq$ overlap (B, A). Table 3 does not report values smaller than 0.1%.

As shown in Table 2, and to the best of our knowledge, this is the first attempt to provide a comprehensive and representative dataset to be used for tackling DGA-based botnets. In using our proposed dataset, one of the main advantages that the research community might acquire relies upon the formal definition (Zago et al., 2019b) of the features and the verifiable feature set obtained as result Zago et al., 2020.

3. UMUDGA: University of Murcia domain generation algorithm dataset

One of the main outcome of this article is the public release of a ML-ready and privacy-aware dataset of AGD. As previously reported in Table 1, our solution matches all the defined properties. To be more precise, as as reported in Section 3 and its subsections, the UMUDGA dataset meets those properties as follows:

- Property 2.1, Synthetic (SYNT) — The dataset is generated by executing malware DGAs and collecting the resulting data, thus achieving the requested SYNT property.
- Property 2.2, General (GNRL) — The dataset includes 38 malware families (Table 4), presenting more than 30 million FQDNs distributed over 50 malware variants besides the legitimate class (Table 2). To the best of our knowledge, this covers the vast majority of publicly known DGA-based malwares.
- Property 2.3 Representative (RPST) — As summarised in Table 4, all variants include at least 10,000 FQDNs (*i.e.*, first Tier in Table 4), having most of them 1 million valid and unique FQDNs (*i.e.*, highest Tier in Table 4).
- Property 2.4, Balanced (BLNC) — As summarised in Table 4, the data are sorted in tiers of different sizes. Within each tier, all the malware variants are fully balanced.
- Property 2.5, Extensible (EXTS) — The code for generating the AGDs is available on Mendeley Data Zago et al., 2020
- Property 2.6, Verifiable (VRFB) — All the data sources are publicly available online. Moreover, the data repository itself reports the formal mathematical definition for each feature presented.
- Property 2.7 Privacy-Orientation (PROR) — The dataset is composed only by *context-free* features, which are natively anonymous and privacy-oriented. They, in fact, do not require any contextual information from the users or the network state (Zago et al., 2019a).
- Property 2.8 Machine Learning Ready (MLRD) — The data have been preprocessed to assure the absence of missing or corrupted data. The repository provides the data in both raw TXT FQDNs lists, CSV and ARFF formats Zago et al., 2020.

Table 4
DGA families.

Tier	Families collected
10,000	CCleaner, Kraken*, Murofet*, Pizd†, Pykspa, SuppoBox*,†, Vawtrak*
50,000	Vawtrak*, Gozi*,†
100,000	Pykspa-noise, QakBot, Ramnit, Tempedreve, Gozi*,†
500,000	Banjori, Murofet*
1,000,000	Alureon, Bedep, ChinAd, CoreBot, CryptoLocker, DirCrypt, Dyre, Fobber*, Kraken*, Locky, Matsnu†, Necurs, Nymaim, PadCrypt, Proslikefan, Pushdo, Qadars, Ramdo, Ranbyus*, Rovnix†, Shiotob, Simda, Sisron, Symmi, Tinba, Vawtrak*, Zeus-NewGoz.

† Wordlist-based family. * Multiple variants.

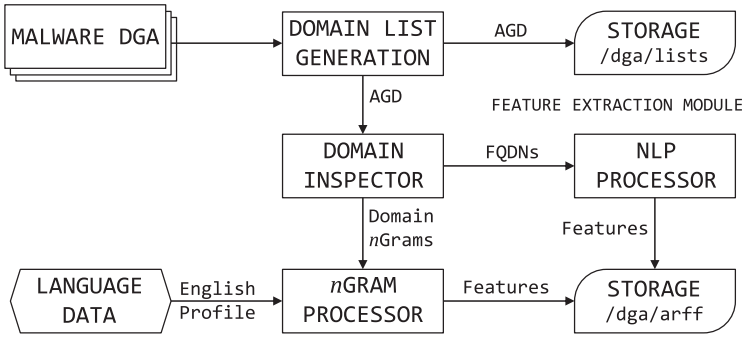


Fig. 3. Architecture for the dataset generation showing both the required inputs (the malware DGA and the English Language Data) and the provided outputs (the AGD lists and the AGD features sets).

- Property 2.9 Labelled (LABL) — The data are available as collection of malware variants data sources, natively tagged with the correct label.

As previously stated (Zago et al., 2019a), the public release of a ML-ready dataset represents an innovative response to a well-known challenge in the cybersecurity field, however, future researches are still required to extend it to both *context-free* and *context-aware* features.

The following Section 3.1 presents the architecture of the data processing and collecting framework, while Section 3.2 will describe the methodology used for designing and building the dataset.

3.1. The UMUDGA architecture

The generation framework flow consists in executing malware DGA to collect the AGD and then process them to obtain the relevant features. Fig. 3 illustrates the data flow.

To be more precise, and as explained in detail in Section 3.2, the raw FQDNs are obtained by executing in a controlled environment the malwares' DGAs, which in turn are both saved as raw lists and processed to become ML-ready data. The AGDs are firstly processed by the `Domain Inspector` procedure, entry point for the `Feature Extraction Module`, which takes care of validating each domain name and extract relative *n*Grams. The validation is carried out with both the Google Guava `InternetDomainName` class (The Guava Authors, 2009) and the Apache Commons Validator library (The Apache Software Foundation, 2017). The former performs syntax validation while the latter evaluates the domain names according to the standards RFC 1034 (Mockapetris, 1987, Section 3) and RFC 1123 (Braden, 1989, Section 2.1). A detailed explanation of the technical validation is offered in a companion article that provides the dataset description (Zago et al., 2020).

Firstly, the domains are analysed by a Natural Language Processing (NLP) procedure that extracts 15 common features such as the length of the domain, its vowel ratio, among others; while, secondly, the *n*Gram are analysed by the corresponding procedure to extract 31 features for each *n*Grams size (with $n = 1, 2, 3$), *e.g.*, entropy, frequencies. As previously mentioned, the features, among other technical aspects of the dataset, are described in Zago et al., 2020.

Table 5
List of features generated by the `NLP Processor` for each FQDN.

Code	Description
`L-`x	String length of x domain level
`N`	Number of domain levels
`LC-C`	Longest consecutive consonance sequence
`LC-D`	Longest consecutive number sequence
`LC-V`	Longest consecutive vowel sequence
`R-CON-`x	Ratio of consonants characters
`R-LET-`x	Ratio of letter characters
`R-NUM-`x	Ratio of numerical characters
`R-SYM-`x	Ratio of symbolical characters
`R-VOW-`x	Ratio of vowel characters

where $x \in \{\texttt{FQDN}, \texttt{2LD}, \texttt{OLD}\}$ denotes the domain levels.

The `nGrams Processor` primarily compares the *n*Grams distribution with the corresponding distribution of the English language, provided by the Leipzig Corpora (Goldhahn et al., 2012).

The following methodology Section (3.2) discuss in detail the content of each module and procedure.

3.2. Methodology for building the UMUDGA dataset

This section aims to illustrate the procedures, the assumptions and the tools used to collect, filter, and generally prepare the data. The process is twofold, *i.e.*, the data are firstly collected in form of raw FQDN lists (see Section 3.2.1); secondly, the domains lists are processed and the resulting features files are saved in the dataset (see Section 3.2.2).

3.2.1. Generation and collection of FQDNs

Malware lists such as Netlab 360, 2019; (Malware domain list, 2009); Bambenek Consulting, 2019; (Risk Analytics, 2007) are quite common and used on daily basis by multiple firewalls and anti-malware providers. However, the provided data are rarely identified with the malware family or variant, thus they are more often labelled as generic threats. As a consequence, the approach that we took for building this dataset is slightly different. That is to say, instead of collecting lists of AGDs from multiple online sources like those, we have been looking for the study and the actual implementation of the malwares' DGAs. Therefore, our data are exclusively generated by executing DGAs implementations in a controlled environment (achieving the Property 2.1, SYNT). The source code has been adapted from three of the main providers of DGA implementations Abakumov (2014); Chaignon (2015); Bader, 2019 and will be released in a public repository (thus achieving Property 2.5, EXTS).

Each algorithm's random function is initialised with a fixed seed and when required, this random module is also used to derive both dates and other arguments. For example, whenever a DGA requires a new date to calculate the corresponding AGDs, the support infrastructure provides a random datetime string derived from the initialisation seed (Property 2.6, VRFB).

The generated AGD are guaranteed to be unique within the class, however, this property is not forced across the dataset when considered as a whole. To be more precise, there are 551 collisions shared among 10 malware variants (*e.g.*, `Pizd` shares 441 AGDs with first version of `SuppoBox`). However, as highlighted by Plohmann et al. (2016) and proved by our analysis (551 over 30.8 million total domains), collisions are quite rare among different malware families, and negligible when considering also the legitimate category. Therefore, removing eventual collisions does not produce a statistically significant change in the malware family distribution, while substantially increasing the quality of the data for ML usage. To be as complete as possible, the full list of colliding FQDN is available in the repository Zago et al., 2020.

Each DGA is executed until it stops generating new domain names or it reaches 1 million AGD (Property 2.3, RPST). Resulting FQDNs are then truncated to the highest completed tier, *i.e.*, 10k, 50k, 100k, 500k, 1M. Table 4 reports the list of the families sorted according to their highest tier, *e.g.*, at least one variant of `Kraken` consists of only 10k AGDs. To the best of our knowledge, these 50 variants are covering the vast majority of known DGA-powered malwares (Property 2.2, GNRL).

To complete the dataset, the last set of FQDNs is obtained by joining the Alexa Internet Inc, 2019 and the Majestic Million (Majestic-12-Ltd, 2019) domain lists. From the two lists, a million unique domains are extracted and allegedly considered as *legitimate*. However, two main problems aroused when validating those domain names, in fact a total amount of 178 FQDNs fail to pass the validation procedure. To be more precise:

- 38 of them use one of the new generic top level domains (gTLDs) which are still not included in the list of accepted gTLDs as per the last update of the validation library (Apache Commons Validator (The Apache Software Foundation, 2017) – v1.6, 04/02/2017). Namely, `.africa` (delegated on 14/02/2017), `.charity` (04/06/2018), `.hotels` (03/04/2017), `.inc` (16/07/2018) and `.sport` (08/01/2018).
- 140 domains are technically invalid because of the presence of at least one underscore character ("_"): the validation library checks the domains against the RFC 1123 (Braden, 1989), which limits host names to letters, digits and hyphen. The policy for the underscore character has been clarified later with the RFC 2181 (Elz and Bush, 1997, Section 11).

3.2.2. Preprocessing and feature extraction

To begin with the inner mechanisms of such module, as illustrated in Figure 3, it is imperative to restate what firstly proposed by Zago et al. (2019a), *i.e.*, the generated features belong to the *Context-Free* family. That is to say, the features are related only to a FQDN and are independent of contextual information.

Firstly, the FQDN lists obtained in the previous steps are analysed and syntactically validated against RFC 1034 (Mockapetris, 1987, Section 3) and RFC 1123 (Braden, 1989, Section 2.1) by the `Feature Extraction Module` reported in Fig. 3. All the invalid domain names are replaced with other unique samples obtained by the corresponding DGA. The dataset documentation provides the required mathematical formalism for each implemented feature Zago et al. (2019b).

With regards to Fig. 3, the first process is the `NLP Processor`, which analyses each domain name as string, with little to none knowledge about its structure as FQDN or its language. The `NLP Processor` extracts 22 features from the domain name, which are listed and described in Table 5. To improve the readability, Table 5 presents a list of meta features, that indicates that a specific mathematical formula has been applied to multiple targets. That is to say, we indicate with x the domain level used as argument for calculating the feature, having x equals to either "`FQDN`" (that stands for Fully Qualified Domain Name), "`2LD`" (that stands for Second Level Domain) and "`OLD`" (that stands for Other Level Domain, which comprehends any domain level below the second).

The second process, namely the *n*`Grams Processor`, is the one that analyses the domain name as a collection of tokens, called *n*Gram. Firstly, each FQDN is divided into chunks of size n and then compared to the reference ones belonging to the English language. These last collections are obtained by preprocessing the Leipzig Corpora (Goldhahn et al., 2012) which includes 1 million words from Wikipedia (2016 update). There are a total of 29 features extracted from such analysis; Table 6 presents them. Despite having these features listed once in Table 6, the dataset includes them applied to 1Grams, 2Grams and 3Grams for a total of 87 features. That is to say, the 29 features are mathematically defined independently from the chosen length (n) of the chunks used for the analysis and thus can be applied to any *n*Grams size.

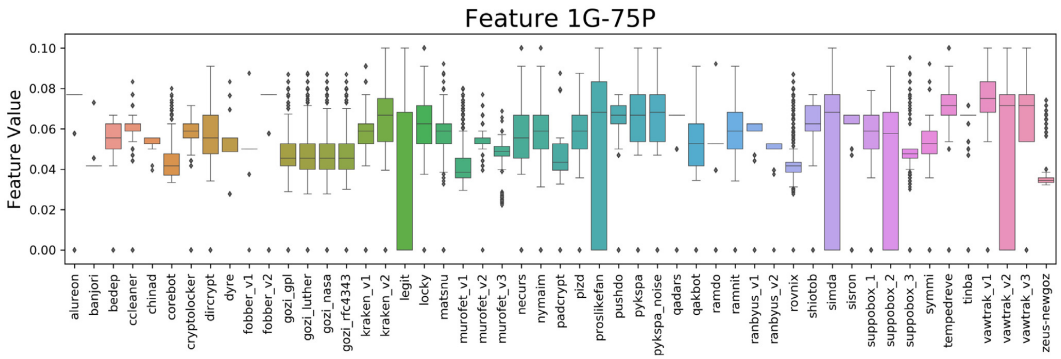
As a side note, some of these features applied to 2Grams and 3Grams are (almost) constant and thus (practically) irrelevant. They are nevertheless included in the dataset for completeness. Fig. 4 presents one of such features, namely the specific case of the 75^{th} percentile of the *n*Gram distributions. For example, Feature `1G-75P` (Fig. 4a) have been proven sufficiently informative for ML applications, despite having its counterparts zeroed-out. In fact, by considering the nature of the feature itself, it does not surprise that both the `2G-75P` (for 2Grams, shown in Fig. 4b) and the `3G-75P` (for 3Grams, shown in Fig. 4c) present a very skewed distributions, where only a few of them have actually a value. Both of them are nevertheless included for symmetry and completeness.

Due to space concerns, the full list of features, with their distributions and descriptions is not included here. Nonetheless, it is publicly available at Zago et al., 2020.

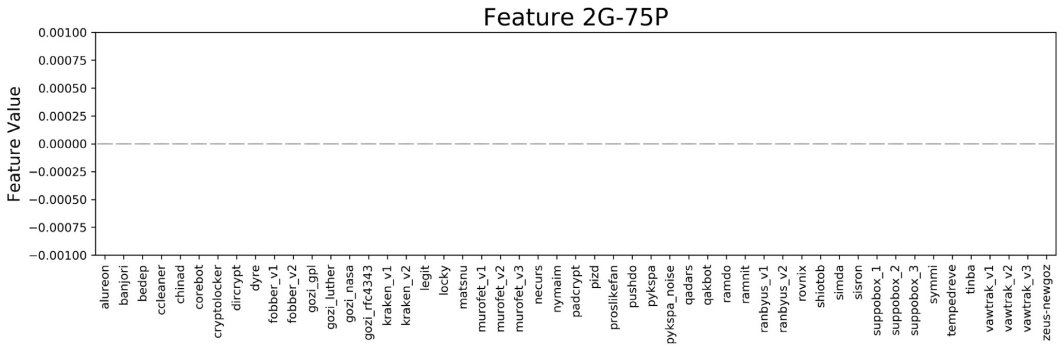
Finally, a survey in terms of where these features originated and when have been used in literature to power ML-based solutions can be found at Zago et al. (2019a).

4. UMUDGA dataset analysis

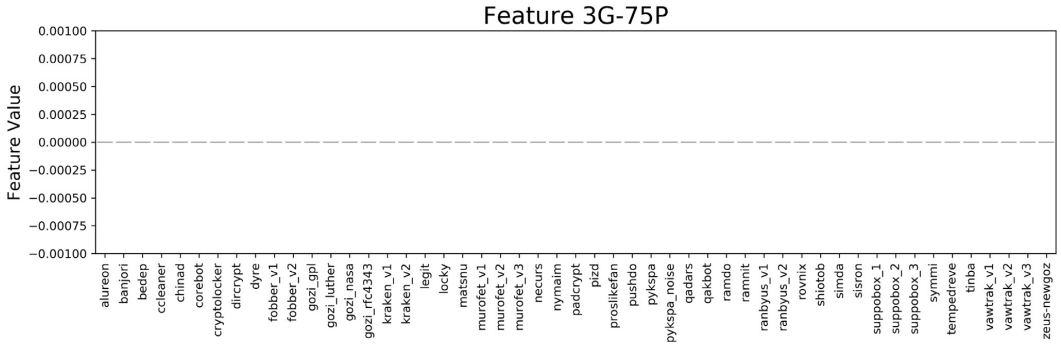
Generally speaking, a first exploratory analysis of the data is suggested and often required before applying more sophisticated ML algorithms. This section aims to provide a brief characterisation of the malware variants and their properties by presenting and discussing the results of a first and naïve analysis. The anal-



(a) Boxplots for the Feature `1G-75P` for the 1Grams distributions.



(b) Boxplots for the Feature `2G-75P` for the 2Grams distributions.



(c) Boxplots for the Feature `3G-75P` for the 3Grams distributions.

Fig. 4. Boxplots comparisons for the Feature *n*`G-75P` that indicates the 75th percentile of the *n*Gram distributions. (See Table 6).

ysis carried out in the section below is performed over the first tier of malwares, as depicted in Table 4.

Among the features described in Section 3, few of them present signs that indicate their low quality with regards to the data analysed.

Firstly, there are features in which nearly all the values are identical, to be more precise the percentiles of frequencies, their mean and median for $n = 1, 2, 3$ are statistically indistinguishable. Moreover, the different ratios are mostly alike when considering the second level domain (2LD) and the other level domain (OLD) parts. Lastly, the number of repeated *n*Gram and their covariance with respect to the $n = 3$ size are also practically identical. Lastly, there are multiple features that are highly correlated, thus considering the ones with a correlation index lower than 0.9.

On the one hand, when considering the structure of the AGD, *i.e.*, most of them are composed only by two domain levels, it is not surprise that the NLP features like the ratios or the lengths

Table 6
List of features generated by the `nGrams Processor` for each *n*Grams set.

Code	Description	Code	Description
nG-DIST	Number of distinct *n*Gram	nG-REP	Number of repeated *n*Gram
nG-25P	25th percentile of frequencies	nG-E	Entropy
nG-50P	50th percentile of frequencies	nG-COV	Covariance[1]
nG-75P	75th percentile of frequencies	nG-KEN	Kendall's Correlation[1]
nG-MEAN	Mean of frequencies	nG-PEA	Pearson's Correlation[1]
nG-QMEAN	Quadratic Mean of frequencies	nG-SPE	Spearman's Correlation[1]
nG-SUMSQ	Squared sum of frequencies	nG-TSUMSQ	Squared sum of target language frequencies[1]
nG-VAR	Variance of frequencies	nG-TVAR	Variance of target language frequencies[1]
nG-PVAR	Population VAR of frequencies	nG-TPVAR	Population VAR of target language frequencies[1]
nG-STD	Standard deviation of frequencies	nG-TSTD	Standard deviation of target language frequencies[1]
nG-PSTD	Population STD of frequencies	nG-TPSTD	Population STD of target language frequencies[1]
nG-SKE	Skewness of frequencies	nG-TSKE	Skewness of target language frequencies[1]
nG-KUR	Kurtosis of frequencies	nG-TKUR	Kurtosis of target language frequencies[1]
nG-PRO	Pronounceability score[1]	nG-TSUM	Sum of target language frequencies[1]
nG-NORM	Normality score	nG-DST-KL	Kullback-Leiber divergence[1]
nG-DST-CA	Canberra Distance[1]	nG-DST-JI	Jaccard Index measure[1]
nG-DST-CH	Chebyshev Distance[1]	nG-DST-EU	Euclidean Distance[1]
nG-DST-EM	Earth Movers Distance[1]	nG-DST-MA	Manhattan Distance[1]

where [1] is about the English language, and nG indicates the size of the *n*Grams collection used for the group of features.

are correlated to each other, specifically when looking at the ones calculated over the FQDN and the 2LD.

However, giving the *n*Grams analysis in combination with the shortness of the domain names, it is not unexpected that the features that are most sensible to zeros have been discarded. For example, the number of distinct *n*Grams, their variance, standard deviation, skewness, etc., presents high correlation grades with each other. Also the Manhattan distance, calculated as the sum of the absolute deviation, tends to be highly correlated with other distances included in the data.

In order to further explore the data we designed two ML classification tasks. These experiments were conducted on a virtual server with 18 cores running at 2.30GHz and 50 GB of DDR3 RAM at 1600 MHz. Experiments were run using Orange3 (Demšar et al., 2013). Six classifiers were applied and cross-validated using a stratified 10-fold approach, using the following configuration and ML techniques:

- **AdaBoost (AB)** — Using 50 trees as base estimators, with SAMME.R classificator (updates base estimatorâ;;s weight with probability estimates) and linear regression loss function.
- **Neural Network (NN)** — Single hidden layer with 100 nodes activated with the Rectified Linear unit (ReLu) function, weight optimised with the stochastic gradient-based optimiser (Adam), $\alpha = 0.0010$ and 200 max iterations.
- **Random Forest (RF)** — Using 10 trees, considering up to five attributes at each split and without splitting subsets smaller than five.
- **Support Vector Machines (SVM)** — Configured with $C = 1.00$, $\epsilon = 0.10$ and using the RBF Kernel.
- **Decision Tree (DT)** — Two minimum instances in leaves, do not split trees smaller than five, having a max depth of 100. Exiting condition when the majority reaches 95%.
- **k-Nearest Neighbours (kNN)** — Five neighbours using the Euclidean metric and a uniform weight.

In the following sections, unless otherwise stated, the experiments were conducted using all the variants belonging to the Tier 10,000, *i.e.*, a balanced dataset with 10,000 samples for each class.

The two classification tasks are defined as follows:

Experiment 1 (Binary)**.** The Binary task is designed to answer the ML question of separating legitimate FQDNs from malicious AGDs, considering all malware families as a single category.

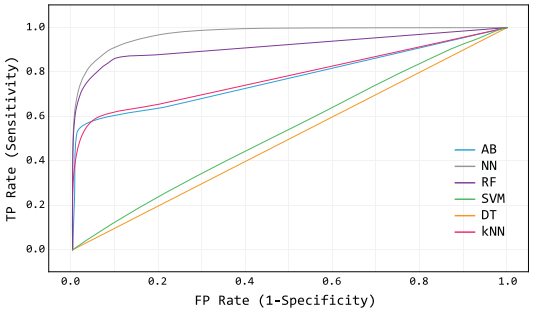
Experiment 2 (Multiclass)**.** The Multiclass task is designed to classify not only the legitimate FQDNs, but also sort malware samples according to their variants.

Figs. 5 and 6 report both the results in terms of classifiers performances and the Receiver Operating Characteristic (ROC) curves for the Binary (Exp. 1) and the Multiclass (Exp. 2) experiments, respectively. It is worth mentioning that the ROC curves are generated considering the legitimate (legit) class as target, thus showing the support in correctly predicting this class on average with respect to the 10 folds analysed. Figs. 5 and 6 a have been simplified by firstly interpolating the values and then by smoothing the curve to reduce the number of points (accepting a loss of 5% precision).

From the two images, it is possible to notice that the results are somewhat different from the high-grade extremely precise results obtained in literature over subsets of the same classes (Zago et al., 2019a). One could argue that the data are different, which is somewhat correct; data sources are different, and rarely publicly shared for subsequent analysis (*i.e.*, Property 2.6, VRFB). We assume that the data obtained from the generators, as described in Section 3, for a specific malware variant are taken from the same space, thus having similar characteristics. It can also be stated that the features used are not the same, nor calculated in the same way. This assertion surely holds. However, as shown in our early research (Zago et al., 2019a), most literature works made usage of context-free features, which have been collected, analysed and re-implemented as presented in Section 3. Moreover, the features selected in the literature are rarely mathematically defined, let alone implemented and made publicly available (*i.e.*, Property 2.5, EXTS). Once again, without having a structural formalism of the performed data processing that led to those results, comparing them is somewhat difficult.

As expected, the Binary experiment (Exp. 1) performs much better than the Multiclass one (Exp. 2). And, to further explore these average results, a sample of a class-specific analysis is proposed in Sections 4.1 and 4.2.

Fig. 7 presents the confusion matrix for the Random Forest classifier in the Multiclass experiment as an heatmap chart. In the figure, the darker the colour, the better is the classifier precision. From the picture, it appears clear that although the classifier achieves excellent results in most of the classes, some clusters of classes appears to be difficult to separate, ultimately causing the degraded overall performances presented in Fig. 6b. In Fig. 7, actual percentages are omitted for clarity, the complete report is available at Zago et al., 2020.

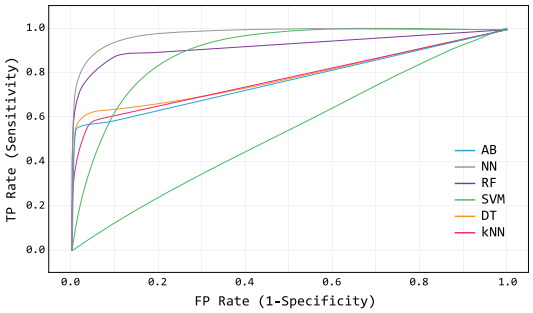


(a) Average ROC curves

Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.981	0.982	0.981	0.770	0.981
NN	0.989	0.988	0.989	0.973	0.988
RF	0.989	0.988	0.989	0.914	0.989
SVM	0.403	0.403	0.965	0.403	0.556
DT	0.980	0.961	0.980	0.500	0.971
kNN	0.986	0.984	0.986	0.781	0.983

(b) Classifiers performances

Fig. 5. Results for the Binary experiment (Exp. 1).



(a) Average ROC curves for the legitimate class

Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.665	0.666	0.665	0.829	0.666
NN	0.772	0.775	0.772	0.993	0.769
RF	0.710	0.706	0.710	0.974	0.704
SVM	0.420	0.432	0.420	0.945	0.385
DT	0.672	0.671	0.672	0.850	0.672
kNN	0.312	0.297	0.312	0.765	0.297

(b) Classifiers performances

Fig. 6. Results for the Multiclass experiment (Exp. 2).

To further explore these classes clusters, we picked two clusters of variants, namely the one formed by `Alureon` and `Fobber (2nd version)` and the one composed by `Bedep`, `DirCrypt` and `Ramnit`. The results of these analysis are presented in Sections 4.1 and 4.2, respectively. The experiments have been performed with amount compatible with the highest tier available for each malware variant. As indicated in both section, a potential solution for this issue might be represented by the double detection process implemented by Gil Pérez et al. (2017). To be more precise, by considering the elements of the cluster as a single unique entity, it is possible to flag as "suspicious" a DNS query made by some user (*i.e.*, the first, high-level detector). A dedicated analyser (*i.e.*, the second, fine-granularity detector) might then take care of performing a more precise, accurate and time consuming analysis of the identified user.

To further explore the potential researches that might spring from the usage of this dataset, it is worth mentioning the challenges related to ML identification of DGA-based botnets. For example, it is unclear whether is possible to define a common signature for AGDs as a group (Exp. 1) or if from a sample analysis is possible to incontrovertibly identify the malware family or even the precise variant (Exp. 2). Further researches are also required in terms of algorithms application to explore the data, for example, literature suggests that both deep learning and clustering solutions might prove useful in identifying known and unknown malwares, respectively.

4.1. *Fobber (2nd version)* versus *Alureon*

The first cluster of errors belongs to `Fobber (2nd version)` and `Alureon`. Both variants' DGAs are publicly available at Zago et al., 2020. The first step is to execute a specific ML experiment oriented toward those two classes, which results are reported in Fig. 8a (`ROC` curve) and Fig. 8b (classifiers performances). From these, it is clear that the feature set chosen, in combination with the data and the algorithms configurations do not permit to separate the two classes.

To further explore the two classes, we have printed their 1Grams distributions in Fig. 8c together with the English distribution and a uniform one. As suggested by Knuth (1997), we performed a Pearson's ChiSquare Test to compare them, and the results are reported in Fig. 8d. The test does not reject the hypothesis that the two variants belong to the same uniform distribution. Follows that both malware variants have achieved to generate AGDs within a uniform distribution, and thus impossible to distinguish with the current feature set.

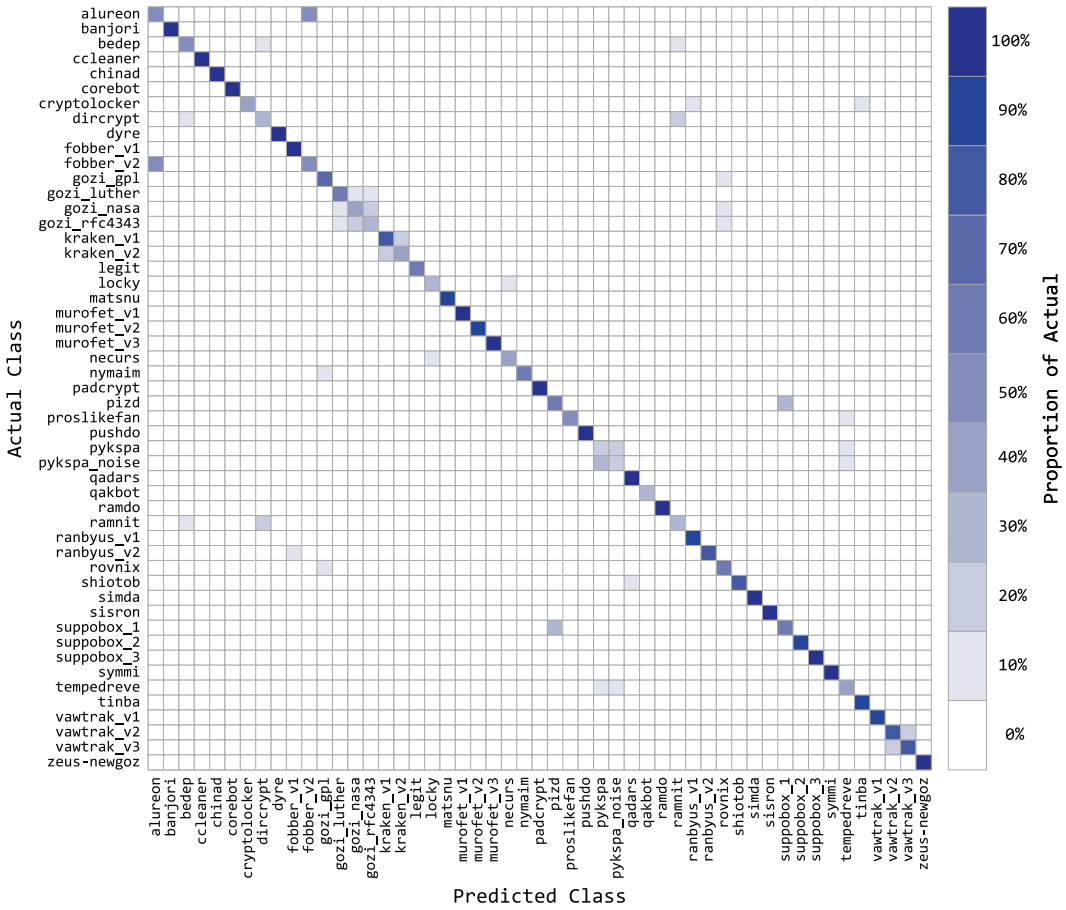


Fig. 7. Confusion Matrix for the Multiclass experiment (Exp. 2, Random Forest).

Further analysis including, but not limited to, context-aware features might result in an effective way to distinguish the two classes. For example, a double cycle detection as proposed by Gil Pérez et al. (2017) might be applied in such case. That is to say, both `Fobber (2nd version)` and `Alureon` can be considered as a single class, and then deeply analysed with a specific component once a detection event occurs. In fact, as shown in both Fig. 9a and b, almost all classifiers achieve extremely high performances while oriented toward distinguishing legitimate domain names from these two joined classes.

4.2. *Bedep* versus *DirCrypt* versus *Ramnit*

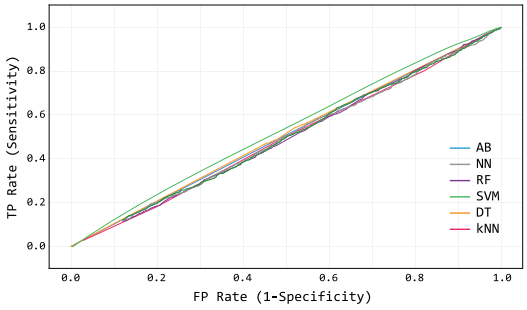
The second cluster of classification errors belongs to three classes, namely `Bedep`, `DirCrypt` and `Ramnit`. However, as shown in Fig. 10 and Fig. 11, it appears that this group is driven by the identical distributions of `DirCrypt` and `Ramnit`, which are then very likely, but not identical to `Bedep`.

The classifiers output for the three malware variants can be found in Fig. 10b, and together with the ROC curve analysis (Fig. 10a) prove that, within this data, the ML algorithms only take educated guesses over the class, without being capable of actually memorising the class characteristics.

The Pearson's ChiSquare test, presented in Fig. 10b, suggests another subgroup formed by `DirCrypt` and `Ramnit`. To analyse and eventually confirm this hypothesis, we conducted a Binary experiment over those two classes, proving that also in this case it is not possible to separate the two classes (Fig. 10c and d).

However, in Fig. 10a it is possible to notice that both `DirCrypt` and `Ramnit` differ from the uniform distribution in a few cases. To be more precise, our implementation of `Ramnit` DGA does not ever produce the letter "`z`" in any domain. This leads to a missing value in the distribution value, which can be mapped as zero. Having one zero in the distribution causes the ChiSquare test to fail for not being defined at zero. This condition, together with the fact that we do not have a specific feature for each character (including "`z`" and "`c`", as highlighted in Fig. 10a) permits to not consider the character during this test. The results are available in Fig. 10b and, as expected, both three variants achieve to be uniform (or almost uniform).

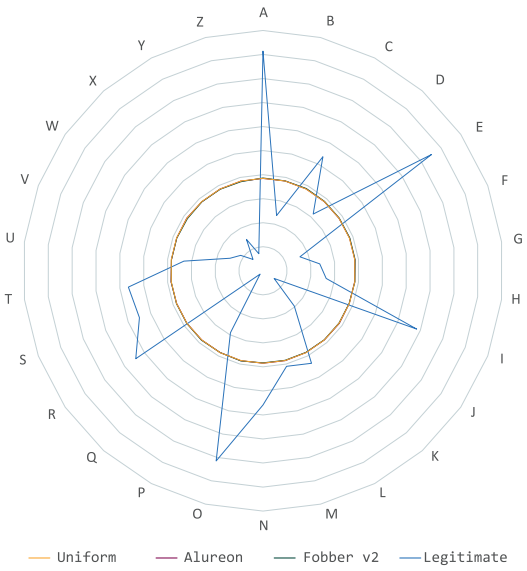
As suggested in Section 3, a double cycle detection (Gil Pérez et al., 2017) might take advantage of other class-specific characteristics to perform a deeper analysis. For instance, Fig. 12 reports the



(a) ROC curve

Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.504	0.504	0.504	0.504	0.504
NN	0.494	0.494	0.494	0.493	0.494
RF	0.503	0.503	0.503	0.505	0.503
SVM	0.501	0.501	0.501	0.499	0.500
DT	0.500	0.500	0.500	0.500	0.500
kNN	0.502	0.501	0.501	0.502	0.501

(b) Classifiers performances



(c) Distributions comparison

Distribution	Expected		
Observed	Uniform	Alureon	Fobber v2
Alureon	1.000	–	0.979
Fobber v2	0.983	0.979	–

(d) Pearson chi-squared test (χ^2)

Fig. 8. Comparative analysis of Fobber (2nd variant) and Alureon.

classifier results and the ROC curve for the six classifiers defined earlier in Section 4. It is clear that most classifiers can achieve extremely good performances, and thus can act as a filter before a more deep and accurate analysis is performed.

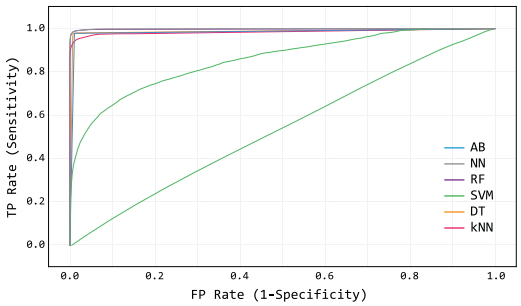
4.3. Discussion

As previously mentioned in Section 4, only a few clusters have been analysed and reported here. The same analysis however has been carried out for all classification errors. To be more precise, we decided to aggregate the classes that have at least 20% of misclassification between each other. For example, Fobber (2nd version) and Alureon share around 50%.

Following what previously suggested throughout this section, an approach like the one defined in Gil Pérez et al. (2017) might be appropriate to develop a full-fledged detection solution for tackling DGA-based botnets. To validate this hint, Experiment 2 have been executed on a tweaked dataset that considers clusters of classes as classification target instead of malware variants. Fig. 13 summarises the results for this scenario and to be more precise, Fig. 13b reports the classifiers performances while Fig. 13a presents a comparison between the previously shown Multiclass experiment's classifiers' F1 scores (Fig. 6b) and the ones obtained by the classifiers in this scenario.

As depicted in Fig. 13a there is a general improvement across all the classifiers, which is also clearly visible in the related confusion matrix available in Fig. 14.

Once a FQDN is flagged as suspicious, for example by classifying it as in the new aggregated class `Alu-Fobv2`, which comprehends both `Fobber (2nd version) and Alureon`, further

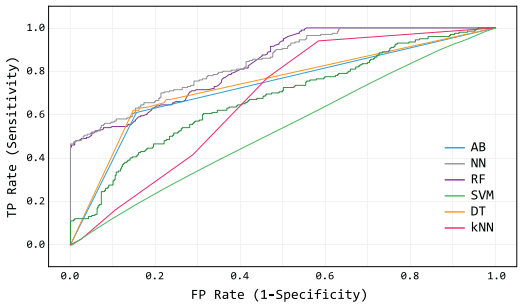


(a) ROC curve for the `legit` class while compared to `Fobber (2nd version)` and `Alureon`

Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.986	0.979	0.978	0.984	0.979
NN	0.990	0.986	0.983	0.999	0.985
RF	0.989	0.986	0.983	0.998	0.984
SVM	0.701	0.736	0.669	0.855	0.701
DT	0.979	0.981	0.977	0.987	0.979
kNN	0.958	0.991	0.927	0.985	0.958

(b) Classifiers performances for the `legit` class while compared to `Fobber (2nd version)` and `Alureon`

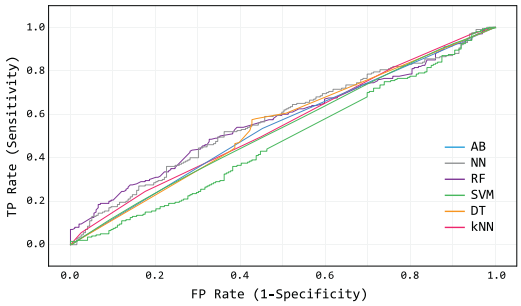
Fig. 9. Comparative analysis of `Fobber (2nd variant)` and `Alureon` as a single class against the `Legit` one.



(a) ROC curve for `Bedep` vs `DirCrypt` vs `Ramnit`

Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.525	0.533	0.525	0.644	0.528
NN	0.518	0.525	0.518	0.735	0.521
RF	0.497	0.499	0.497	0.722	0.499
SVM	0.400	0.389	0.400	0.615	0.389
DT	0.522	0.519	0.522	0.656	0.519
kNN	0.425	0.410	0.425	0.605	0.410

(b) Classifiers performances for `Bedep` vs `DirCrypt` vs `Ramnit`

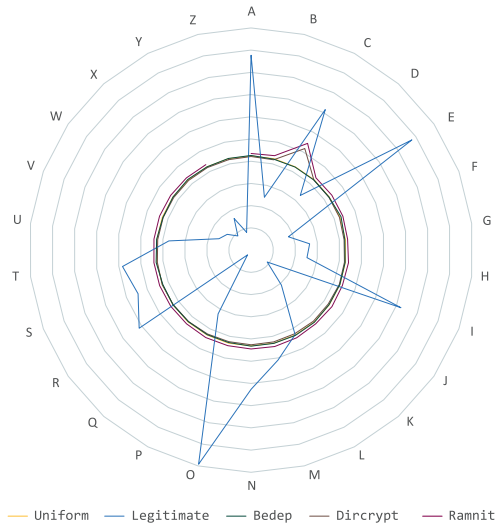


(c) ROC curve for `DirCrypt` vs `Ramnit`

Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.525	0.525	0.525	0.549	0.525
NN	0.552	0.553	0.552	0.548	0.552
RF	0.515	0.515	0.515	0.478	0.514
SVM	0.555	0.555	0.555	0.576	0.555
DT	0.557	0.558	0.557	0.572	0.557
kNN	0.542	0.543	0.542	0.542	0.542

(d) Classifiers performances for `DirCrypt` vs `Ramnit`

Fig. 10. Comparative analysis of `Bedep`, `DirCrypt` and `Ramnit` – Part 1.

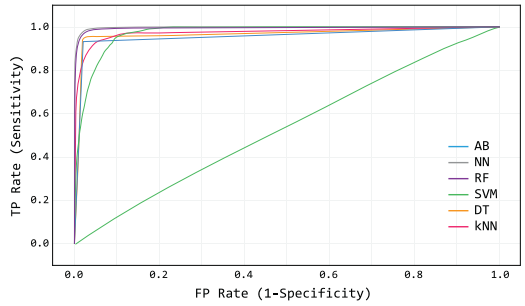


(a) Distributions comparison

Distribution	Expected			
Observed	Uniform	Bedep	DirCrypt	Ramnit[1]
Bedep	1.000	–	0.000	0.000
DirCrypt	0.972	0.993	–	1.000
Ramnit[1]	0.929	0.991	1.000	–

(b) Pearson chi-squared test (χ^2), where [1] indicates that the tests have been conducted excluding the character "z"

Fig. 11. Comparative analysis of `Bedep`, `DirCrypt` and `Ramnit` – Part 2.



(a) ROC curve for the `legit` class while compared to `Bedep`, `DirCrypt` and `Ramnit`

Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.968	0.938	0.932	0.956	0.935
NN	0.980	0.956	0.965	0.998	0.961
RF	0.977	0.944	0.964	0.994	0.954
SVM	0.838	0.834	0.438	0.973	0.575
DT	0.970	0.933	0.947	0.965	0.940
kNN	0.946	0.887	0.898	0.974	0.892

(b) Classifiers performances for the `legit` class while compared to `Bedep`, `DirCrypt` and `Ramnit`

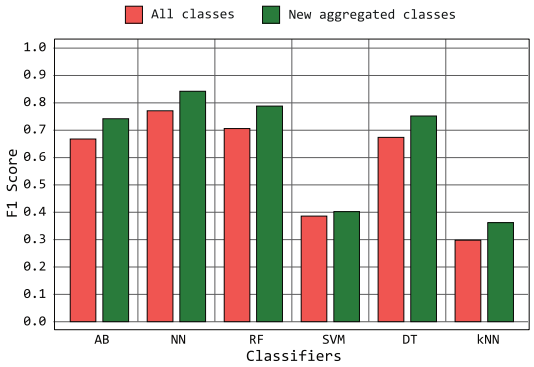
Fig. 12. Comparative analysis of `Bedep`, `DirCrypt` and `Ramnit` versus the legitimate class.

analysis can be deployed to perform deep inspection that might allow to pinpoint the exact malware variant, thus enabling the deployment of appropriate countermeasures.

Future researches are, however, required to establish whether an improved *context-free* set of features may be able to distinguish between the clusters of classes or if a deeper analysis based on *context-aware* features is required. Nevertheless, this result, along with the one reported in Fig. 5, validates the concept of having a first, high-level filter for detecting AGDs, followed by a malware-specific intrusive inspection technique.

One could argue that this approach, does not rely on studying and developing a ML approach suitable for solving the previously detailed Multiclass experiment (Exp. 2). That would be correct to claim if the target of this article would have included the proposal for a ML-powered detection framework. Nevertheless, in this context, where the proposal is a dataset to enable such analysis and comparison, the above mentioned claim does not hold. Once again, one of the main contributions of this article is to propose a publicly available dataset that can be used to research, study and deploy new comparable ML-based solutions that do not require harming the users' privacy.

As pinpointed previously by the authors (Zago et al., 2019a), the task of detecting DGA-based botnets without privacy breaches remains an open challenge, especially with the approach of encrypted DNS (Patsakis et al., 2020). Even if some recent studies suggest that Deep Learning (DL) techniques might provide some advantages (Liang and Yan, 2019; Qiao et al., 2019; Vinayakumar et al., 2019), however, a comparison between these solutions is far



(a) F1 Score comparison between the results proposed in Figure 6b and the ones proposed in Figure 13b.

Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.739	0.741	0.739	0.866	0.740
NN	0.847	0.845	0.847	0.995	0.840
RF	0.794	0.788	0.794	0.979	0.788
SVM	0.445	0.456	0.445	0.945	0.456
DT	0.751	0.750	0.751	0.886	0.750
kNN	0.382	0.358	0.382	0.792	0.358

(b) Classifiers performances

Fig. 13. Results for the Multiclass experiment (Exp. 2) after remapping the classes as described in Section 4.3.

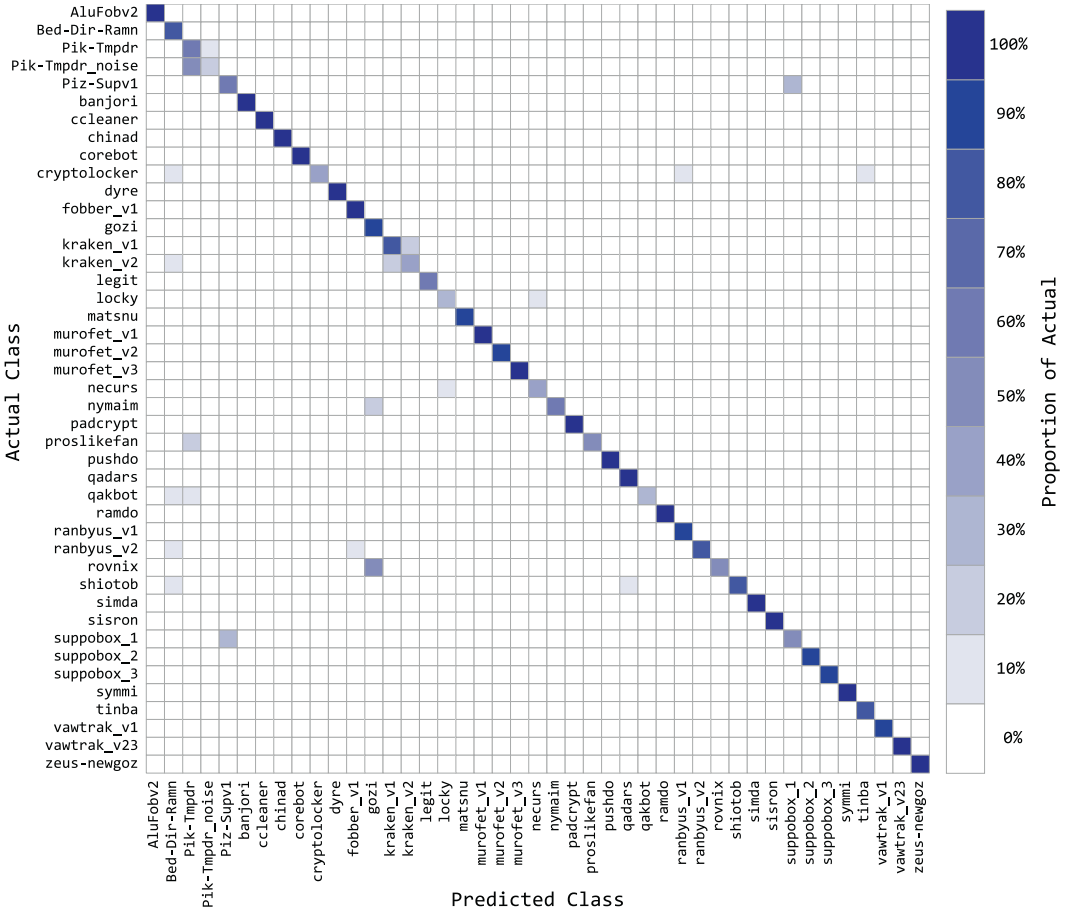


Fig. 14. Confusion Matrix for the Multiclass experiment (Exp. 2, Random Forest) with tweaked data as described in Section 4.3.

to be achievable due to the lack of structured and publicly accessible data.

As previously suggested, with UMUDGA we aim to provide data to overcome the lack of ML-ready and publicly available datasets. However, we acknowledge that our proposal of a context-free dataset is just one side of the problem (Zago et al., 2019a) and the study of the state-of-the-art in terms of context-aware features might result in potential game-changing applications. Despite the unquestionable benefits that supplying fresh data to the scientific community produces, there are several areas at which potential future researchers might look. To cite one, ML techniques have been only used to scratch the surface of the problem (Zago et al., 2019a) and further analysis might lead to innovative products.

Finally, we once again would like to remark that, generally, literature solutions are not replicable, let alone deployable in a real world environment (Zago et al., 2019a). To reach such remarkable objective, future works must:

1. publish the data, a necessary condition to enable the reproducibility of the experiments but also to enable third party future researches to not start from scratch;
2. precisely discuss about initial configurations and subsequent optimisation of applied techniques, including ML algorithms;
3. identify the architecture, the workflow, the environment, the experiments configurations and, in general, provide all the information required to independently redeploy the scenarios and verify the results; and, finally
4. compare the obtained results with the state-of-the-art techniques using reproducible means based, at least, on comparable, if not identical, data sources.

5. Conclusions

Recent technical reports suggest an increasing interest in ML solutions for cybersecurity, and, although sold as an all-comprehensive panacea, their applications are non-specialist at best. Literature researches show a plethora of shady solutions that claim to achieve almost perfect performances without providing enough means to validate, let alone reproduce, the results. The first and foremost key issue regarding this problem is attributable to the data sources, which are not adequately organised or carefully reviewed. In fact, most of the publicly available datasets suffer from important shortcomings that prevent to achieve the required rigorousness, reproducibility and credibility of the research. To the best of our knowledge, Section 2, summarised in Table 1, highlights the well-known properties of the current state-of-the-art in terms of data sources, providing a precise categorisation that may prove useful to future researchers.

As a consequence, we propose our dataset, the University of Murcia Domain Generation Algorithm Dataset (UMUDGA) UMUDGA available at Mendeley Data (Zago et al., 2020), that ultimately achieves all these established properties alongside with a formal and rigorous mathematical data definition (Zago et al., 2019b).

At the same time, it holds that several challenges are yet to be solved. Further researches are required to address the problem of DGA-based botnets. Unlike the related works analysed in Section 2, UMUDGA aims to address the first of the shortcomings of comparable ML results, *i.e.*, the data source.

Finally, the exploratory analysis shows that data manipulation can easily lead to significant improvements in the performances of any ML solutions, and thus should be strictly documented and justified. Moreover, scientists and future researches should transparently adhere to an experimental protocol that follows predetermined and well-established guidelines, which, to the best of our knowledge, nowadays do not exist. Our proposed guidelines aim to serve as a catalyst for creating a standard protocol for ML solutions in network cybersecurity.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work has been supported by a predoctoral and a postdoctoral INCIBE (Spanish National Cybersecurity Institute) grants within the “Ayudas para la Excelencia de los Equipos de Investigación Avanzada en Ciberseguridad” (“Grants for the Excellence of Advanced Cybersecurity Research Teams”) Program, with codes INCIBEI-2015-27353 and INCIBEI-2015-27352, respectively.

References

Abakumov, A., 2014. Andrewaeva/DGA. URL https://github.com/andrewaeva/DGA.
Alenazi, A., Traore, I., Ganame, K., Woungang, I., 2017. Holistic model for HTTP botnet detection based on DNS traffic analysis. In: Proceedings of the 1st International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments, pp. 1–18. doi:10.1007/978-3-319-69155-8_1.
Alexa Internet Inc., 2019. Alexa Top Domains. URL https://www.alexa.com/topsites.
Bambenek Consulting, 2019. OSINT DGA list. URL http://osint.bambenekconsulting.com/feeds/.
Beigi, E.B., Jazi, H.H., Stakhanova, N., Ghorbani, A.A., 2014. Towards effective feature selection in machine learning-based botnet detection approaches. In: Proceedings of the IEEE Conference on Communications and Network Security, pp. 247–255. doi:10.1109/CNS.2014.6997492.
Berman, D., Buczak, A., Chavis, J., Corbett, C., 2019. A survey of deep learning methods for cyber security. Information 10 (4), 1–35. doi:10.3390/info10040122.
Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K., 2015. Towards generating real-life datasets for network intrusion detection. I. J. Netw. Secur. 17, 683–701. doi:10.6633/IJNS.201511.17(6).05.
Braden, R., 1989. Requirements for internet hosts - application and support. STD, 3. RFC Editor doi:10.17487/RFC1123.
Brandt, A., Cove, B., Yu, C., Wisniewski, C., Szappanos, G., Chandraiah, J., Zhang, J., Levy, J., Kohli, P., MacKenzie, P., Cohen, R., Yu, R., Shevchenko, S., Easton, T., 2018. Sophoslabs 2019 treath report. Technical Report. SOPHOS Ltd. URL https://www.sophos.com/en-us/medialibrary/PDFs/technical-papers/sophoslabs-2019-threat-report.pdf.
Bader, J., 2019. BaderJ - Domain generation algorithm. URL https://github.com/baderj/domain_generation_algorithms.
Chaignon, P., 2015. Pchaigno/DGA_Collection. URL https://github.com/pchaigno/dga-collection.
Demšar, J., Curk, T., Erjavec, A., Gorup, Č., Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., Štajdohar, M., Umek, L., Žagar, L., Žbontar, J., Žitnik, M., Zupan, B., 2013. Orange: data mining toolbox in python. J. Mach. Learn. Res. 14 (1), 2349–2353. URL http://jmlr.org/papers/v14/demsar13a.html.
Elz, R., Bush, R., 1997. Clarifications to the DNS specification. RFC, 2181. RFC Editor doi:10.17487/RFC2181.
Eremin, A., 2018. What are botnets downloading? Statistics for the past year on files downloaded by botnets. Technical Report. Kaspersky Labs. URL https://securelist.com/what-are-botnets-downloading/87658/.
Etaher, N., Weir, G.R.S., Alazab, M., 2015. From ZeuS to Zitmo: trends in banking malware. In: 2015 IEEE Trustcom/BigDataSE/ISPA, pp. 1386–1391. doi:10.1109/Trustcom.2015.535.
Fireeye Mandiant Services, 2019. M-Trends 2019 special report. Technical Report. Fire Eye, Inc. URL https://www.fireeye.com/current-threats/annual-threat-report.html.
García, S., Grill, M., Stiborek, J., Zunino, A., 2014. An empirical comparison of botnet detection methods. Comput. Secur. 45, 100–123. doi:10.1016/j.cose.2014.05.011.
Gerald Combs, 1998. Wireshark. URL https://www.wireshark.org.
Gil Pérez, M., Huertas Celdrán, A., Ippoliti, F., Giardina, P.G., Bernini, G., Alaez, R.M., Chirivella-Perez, E., García Clemente, F.J., Martínez Pérez, G., Kraja, E., Carrozzo, G., Alcaraz Calero, J.M., Wang, Q., 2017. Dynamic reconfiguration in 5G mobile networks to proactively detect and mitigate botnets. IEEE Internet Comput. 21 (5), 28–36. doi:10.1109/MIC.2017.3481345.
Goldhahn, D., Eckart, T., Quasthoff, U., 2012. Building large monolingual dictionaries at the leipzig corpora collection: from 100 to 200 languages. In: Proceedings of the 8th International Conference on Language Resources and Evaluation. European Languages Resources Association (ELRA), pp. 759–765.
Hammi, B., Zeadally, S., Khatoun, R., 2019. An empirical investigation of botnet as a service for cyberattacks. Trans. Emerg. Telecommun. Technol. 30 (3), 1–11. doi:10.1002/ett.3537.
Kent, A.D., 2015a. Comprehensive, multi-source cyber-security events. Los Alamos National Laboratory doi:10.17021/1179829.

Kent, A.D., 2015b. Cybersecurity data sources for dynamic network research. Dynamic Networks in Cybersecurity. Imperial College Press, pp. 27–65. doi:10.1142/9781786340757_0002.

Knuth, D.E., 1997. The art of computer programming: seminumerical algorithms, 2, 3rd Addison-Wesley Longman Publishing Co., Inc. doi:10.1137/1012065.

Kührer, M., Rossow, C., Holz, T., 2014. Paint it black: evaluating the effectiveness of malware blacklists. In: Research in Attacks, Intrusions and Defenses, pp. 1–21. doi:10.1007/978-3-319-11379-1_1.

Kujawa, A., Zamora, W., Umawing, J., Segura, J., Tsing, W., Arntz, P., Boyd, C., 2019. 2019 state of malware. Technical Report. Malwarebytes LABS. URL https://resources.malwarebytes.com/resource/2019-state-malware-malwarebytes-labs-report/.

Liang, Y., Yan, X., 2019. Using Deep Learning to Detect Malicious URLs. In: Proceedings of the IEEE International Conference on Energy Internet (ICEI), pp. 487–492. doi:10.1109/ICEI.2019.00092.

Ma, J., Saul, L.K., Savage, S., Voelker, G.M., 2009. Identifying suspicious URLs: an application of large-scale online learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 681–688. doi:10.1145/1553374.1553462.

Malware domain list, 2009. Malware domain list. URL https://www.malwaredomainlist.com/mdl.php.

Mockapetris, P., 1987. Domain names - concepts and facilities. STD, 13. RFC Editor doi:10.17487/RFC1034.

Moustafa, N., 2017. Designing an online and reliable statistical anomaly detection framework for dealing with large high-speed network traffic. Ph.D. thesis. The University of New South Wales.

Majestic-12-Ltd, 2019. The Majestic Million. URL https://majestic.com/reports/majestic-million.

O'Gorman, B., Wueest, C., O'Brien, D., Cleary, G., Lau, H., Power, J.-P., Corpin, M., Cox, O., Wood, P., Wallace, S., 2019. Internet security threat report. Technical Report. Symantec Corporation. URL https://www.symantec.com/security-center/threat-report.

Netlab 360, 2019. DGA families. URL http://data.netlab.360.com/dga/.

Parkour, M., 2015. Contagio malware dump–collection of Pcap files from malware analysis. URL http://contagiodump.blogspot.com/2013/04/collection-of-pcap-files-from-malware.html

Patsakis, C., Casino, F., Katos, V., 2020. Encrypted and covert DNS queries for botnets: challenges and countermeasures. Comput. Secur. 88, 101614. doi:10.1016/j.cose.2019.101614.

Plohmann, D., Yakdan, K., Klatt, M., Bader, J., Gerhards-Padilla, E., 2016. A comprehensive measurement study of domain generating malware. In: Proceedings of the 25th USENIX Conference on Security Symposium, pp. 263–278. URL http://dl.acm.org/citation.cfm?id=3241094.3241115.

Putman, C.G.J., Abhishta, Nieuwenhuis, L.J.M., 2018. Business model of a botnet. In: Proceedings of the 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing, pp. 441–445. doi:10.1109/PDP2018.2018.00077.

Qiao, Y., Zhang, B., Zhang, W., Sangaiah, A.K., Wu, H., 2019. DGA domain name classification method based on long short-term memory with attention mechanism. Appl. Sci. 9 (20), 4205. doi:10.3390/app9204205.

Risk Analytics, 2007. DNS-BH - Malware domain blocklist. URL http://www.malwaredomains.com.

Sharma, R., Singla, R.K., Guleria, A., 2018. A new labelled flow-based DNS dataset for anomaly detection: PUF dataset. Procedia Comput. Sci. 132, 1458–1466. doi:10.1016/J.PROCS.2018.05.079.

Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A.A., 2012. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Comput. Secur. 31 (3), 357–374. doi:10.1016/j.cose.2011.12.012.

Spamhaus Project, 2019a. Spamhaus botnet threat report 2019. Technical Report. Spamhaus Project. URL https://www.spamhaustech.com/botnet-threat-report-2019/.

Spamhaus Project, 2019b. Spamhaus botnet threat update: Q1-2019. Technical Report. Spamhaus Project. URL https://www.spamhaus.org/news/article/784/spamhaus-botnet-threat-update-q1-2019.

Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A., 2009. A detailed analysis of the KDD CUP 99 data set. In: Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6. doi:10.1109/CISDA.2009.5356528.

The Apache Software Foundation, 2017. Apache commons validator. URL https://commons.apache.org/proper/commons-validator/.

The Guava Authors, 2009. Google Guava - InternetDomainName class. URL https://github.com/google/guava/wiki/InternetDomainNameExplained.

Vinayakumar, R., Soman, K.P., Poornachandran, P., 2018. Detecting malicious domain names using deep learning approaches at scale. J. Intell. Fuzzy Syst. 34 (3), 1355–1367. doi:10.3233/JIFS-169431.

Vinayakumar, R., Soman, K.P., Poornachandran, P., Akarsh, S., Elhoseny, M., 2019. Improved DGA Domain Names Detection and Categorization Using Deep Learning Architectures with Classical Machine Learning Algorithms. Springer International Publishing, pp. 161–192. chapter 8.

Vormayr, G., Zseby, T., Fabini, J., 2017. Botnet communication patterns. IEEE Commun. Surv. Tutor. 19 (4), 2768–2796. doi:10.1109/COMST.2017.2749442.

Wylie, B., 2013. SuperCowPowers - data hacking. URL https://github.com/SuperCowPowers/data_hacking/tree/master/dga_detection.

Zago, M., Pérez, M., Martínez Pérez, G., 2019a. Scalable detection of botnets based on dga: Efficient feature discovery process in machine learning techniques. Soft Computing.

Zago, M., Gil Pérez, M., Martínez Pérez, G., 2019b. UMUDGA: a dataset for profiling algorithmically generated domain names in botnet detection. Data in Brief. [Submitted with this manuscript].

Zago, M., Gil Pérez, M., Martínez Pérez, G., 2020. UMUDGA: University of Murcia domain generation algorithm dataset. Mendeley Data doi:10.17632/y8ph45msv8.1.

Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., Ghorbani, A., Garant, D., 2013. Botnet detection based on traffic behavior analysis and flow intervals. Comput. Secur. 39, 2–16. doi:10.1016/j.cose.2013.04.007.

Zago, M., Gil Pérez, M., Martínez Pérez, G., 2020. UMUDGA: Wiki. Github. URL: https://github.com/Cyberdefence-Lab-Murcia/UMUDGA/wiki. doi: 10.5281/zenodo.3618221.

Mattia Zago is a Ph.D. student at the Department of Information Engineering and Communications of the University of Murcia, Spain. He holds a Master Degree in Computer Science and Engineering (Hons.) from the University of Verona, Italy. His research focuses on information & communication systems security and artificial intelligence, with particular dedication to machine learning applications to network intrusion detection and response systems and behavioural modelling. More info at https://webs.um.es/mattia.zago.

Manuel Gil Pérez is Assistant Professor in the Department of Information Engineering and Communications of the University of Murcia, Spain. His research focuses on cybersecurity, intrusion detection systems, trust management, privacy-preserving data sharing, and security operations in highly dynamic scenarios. He is co-author of more than 60 scientific publications in journals, magazines and conference papers. He received M.Sc. and Ph.D. degrees (Hons.) in computer science from the University of Murcia. More info at https://webs.um.es/mgilperez.

Gregorio Marínez Pérez is Full Professor in the Department of Information Engineering and Communications of the University of Murcia, Spain. His scientific activity is mainly devoted to cybersecurity, privacy and networking. He is working on different national and European IST research projects (25 in the last decade) on these topics, being principal investigator in most of them. He has published 160+ papers in national and international conference proceedings, magazines and journals. More info at https://webs.um.es/gregorio.

3

Early DGA-based botnet identification

Title:	Early DGA-based botnet identification: *pushing detection to the edges*
Authors:	Mattia Zago, Manuel Gil Pérez, Gregorio Martínez Pérez
Journal:	Cluster Computing
J.I.F.:	3.458 Q1 (2019)
Publisher:	Springer
Volume:	—
Pages:	—
Year:	2021
Month:	—
DOI:	10.1007/s10586-020-03213-z
Status:	Published

Abstract

With the first commercially available 5G infrastructures, worldwide's attention is shifting to the next generation of theorised technologies that might be finally deployable. In this context, the cybersecurity of edge equipment and end-devices must be a top priority as botnets see their spread remarkably increase. Most of them rely on algorithmically generated domain names (AGDs) to evade detection and remain shrouded from intrusion detection systems, via the so-called Domain Generation Algorithm (DGA). Despite the issue, by applying concepts such as distributed computing and federated learning, the cybersecurity community has prototyped and developed dynamic and scalable solutions that leverage the increased capabilities and connectivity of edge devices. This article proposes a lightweight and privacy-preserving framework that pushes the intelligence modules to the edges aiming to achieve early DGA-based botnet detection in mobile and edge-oriented scenarios. Experimental results prove the deployability of such architecture at all levels, including resource-constrained end-devices.

Keywords

Domain Generation Algorithm (DGA) · Machine Learning · 5G · Cybersecurity · Edge Artificial Intelligence · Federated Learning

Cluster Computing
https://doi.org/10.1007/s10586-020-03213-z

Early DGA-based botnet identification: pushing detection to the edges

Mattia Zago[1] · Manuel Gil Pérez[1] · Gregorio Martínez Pérez[1]

Received: 1 August 2020 / Revised: 9 November 2020 / Accepted: 13 November 2020

Abstract
With the first commercially available 5G infrastructures, worldwide's attention is shifting to the next generation of theorised technologies that might be finally deployable. In this context, the cybersecurity of edge equipment and end-devices must be a top priority as botnets see their spread remarkably increase. Most of them rely on algorithmically generated domain names (AGDs) to evade detection and remain shrouded from intrusion detection systems, via the so-called Domain Generation Algorithm (DGA). Despite the issue, by applying concepts such as distributed computing and federated learning, the cybersecurity community has prototyped and developed dynamic and scalable solutions that leverage the increased capabilities and connectivity of edge devices. This article proposes a lightweight and privacy-preserving framework that pushes the intelligence modules to the edges aiming to achieve early DGA-based botnet detection in mobile and edge-oriented scenarios. Experimental results prove the deployability of such architecture at all levels, including resource-constrained end-devices.

Keywords Domain Generation Algorithm (DGA) · Machine learning · 5G · Cybersecurity · Edge artificial intelligence · Federated learning

1 Introduction

As predicted [5], the past five years have seen the exponential growth of the research interest in 5G technology, and, nowadays, the first 5G commercial infrastructures are being deployed worldwide. Among other aspects and together with the newly improved service delivery requirements, the ultra-densification of connected devices forces scenarios in which fixed network architectures are not an option anymore [6]. Despite the numerous challenges that remain open [7], the research community has started to look beyond 5G [25].

A critical lesson that the community learned from the 5G research is that the service layer can be decoupled from the network architecture, resulting in frameworks that feature dynamic capabilities such as self-configuration, on-demand scalability, and self-protection. In such a scenario, Artificial Intelligence (AI) can be seen as a necessary construction block to sustain this required dynamicity. Indeed, the enabling technology that can offer the capabilities mentioned above consists of using the AI—Machine Learning (ML) and Deep Learning (DL)—to automate networks and services virtualisation, *e.g.*, AI for software-defined network (SDN) and network functions virtualization (NFV) self-optimisation. As a point of fact, several projects and vendors (5G America [4] and SELFNET [10] among others) proved that intelligent and scalable platforms based on ML-powered SDN/NFV could meet the exponentially increasing demands.

Not surprisingly, with this automation grade, the cybersecurity feature as a key principle across the heterogeneous solutions. Even though numerous frameworks have been developed to provide the required automation, security aspects are often overlooked [7]. To provide an example, proactive and reactive Intrusion Detection Systems (IDSs) appear to be limited to supervised analysis of network flows [7, 15, 24].To be precise, and to the best of our knowledge, there are a few high-grade network IDSs

✉ Mattia Zago
mattia.zago@um.es

Manuel Gil Pérez
mgilperez@um.es

Gregorio Martínez Pérez
gregorio@um.es

[1] Department of Information and Communications Engineering, University of Murcia, 30100 Murcia, Spain

Published online: 02 January 2021

that include machine learning capabilities for other purposes than analysing network flows [7]. However, a clear trend is the widespread adoption of ML solutions at the farthest edges of the networks. Indeed, the concept of pushing computation towards the users is considered as one of the enabling technologies of the 21st century [12, 26]. To be precise, both the concepts of AI-for-Edge and AI-on-Edge [12] have been widely explored before under different names, such as Multi-access Edge Computing (MEC), Mobile Cloud Computing (MCC), Transparent Computing (TC), Fog Computing, cloudlet [26]. All of them employed virtualisation techniques (*e.g.*, SDN/NFV) to decouple the services from the underlying hardware and features AI to optimise them automatically.

Notwithstanding the deployed analysis techniques, researchers agree on the urgency of tackling botnets and Advanced Persistant Threats (APTs), and specifically to identify their communication channels before the actual attacks can take place [21, 37, 39]. Within this context, the usage of Domain Generation Algorithms (DGAs) appeared as a clear trend due to the asymmetrical efforts required to sinkhole the generated domains [37, 39]. The paper at hand will focus on ML applications for tackling these communication channels by providing the means to identify DGA-based botnets at scale.

The subject of detecting algorithmically generated domains (AGDs) is, in fact, offering a fertile research topic from multiple standpoints:

(i) firstly, the inherit randomness of the Fully Qualified Domain Names (FQDNs) makes static rule-based IDSs ineffective in favour of ML solutions;
(ii) secondly, the amount of FQDNs to be analysed compels researchers to develop scalable architectures;
(iii) thirdly, browsing history is to be considered a private subject, thus requiring privacy-aware solutions; and,
(iv) finally, it has been proved that APTs can stay dormant for prolonged periods before performing malicious actions, thus compelling preemptive detection capabilities.

In summary, the proposed approach leverages the well-known ideas that support the SDN/NFV paradigm to provide automated and scalable detection and reaction capabilities—also known as security-as-a-service (SECaaS) [40]. To be precise, the proposed framework demonstrates the capabilities of DGA-based botnet detection services deployed on the farthest edges of the network.

Hence, the fivefold contributions of the paper at hand can be outlined as follows.

- Firstly, this research will identify and discuss the key principles of Edge AI applied to ML-based network security;
- secondly, the literary works on DGA-based botnet detection are examined and mapped to the architectural designs that characterise Edge AI, eventually providing a comparison with pros and cons for each architectural design;
- thirdly, after a constructive discussion regarding the SECaaS deployment location, an experimental framework architecture is drafted;
- fourthly, a set of experiments will prove both the soundness of the Edge AI approach and the effectiveness of lightweight and explainable traditional ML algorithms in identifying DGA-based botnets; and,
- finally, the key ideas and principles identified throughout the research are blended in a lesson learned and future work discussion.

The rest of the paper at hand is structured as follows. Firstly, Sect. 2 will report the necessary background in terms of both Edge AI architectures and applications, with a specific focus on the difference between deployment locations; secondly, Sect. 3 will present the proposed prototypical framework for DGA-based botnet detection on edge. Then, Sect. 4 will gather and discuss some critical aspects with particular attention to future research objectives, while, finally, Sect. 5 will provide a conclusive summary.

2 Edge AI to look beyond 5G

To look beyond 5G technology is necessary to consider the enabling technologies that make the 5G ecosystem working [1]. Among them, the most important ones are undoubtedly the SDN, the NFV, the orchestration frameworks, and the containerisation theory [7]. Although Beyond 5G (B5G) heavily relies on wireless technology improvements [42], they are out of the scope for this research. Hence, instead of focusing on what makes B5G possible hardware-wise, this research looks at what architectural solutions and frameworks can be used.

Across the scale, the trend is clear: AI can and should automate and optimise most of these steps [12].

In this research, the main focus is on the early detection of DGA-based botnets in scenarios that features large volumes of data and a high degree of user mobility. As previously shown by [37], both classical ML (*i.e.*, those algorithms and models that do not employ neural networks) and DL solutions have been deployed to tackle this malware threat.

In the context of 5G and B5G networks, where potentially billions of devices are susceptible to malware infections, the ability to detect DGA-based botnets before they activate to attack other systems is critical. Edge AI offers a promising set of tools to study, design, and deploy scalable ad effective detection solutions.

Hence, using the DGA-based botnet detection as a use case, the remaining of this section is structured to introduce the necessary background on Edge AI (Sect. 2.1), the different architectural designs (Sect. 2.2), and their relations with the cybersecurity aspects (Sect. 2.3).

2.1 On the subject of Edge AI

As thoroughly surveyed by several authors [12, 26, 33, 43], the Edge AI discipline encompasses all those techniques intended to move the ML (with great focus on DL ones) towards the end users instead of the cloud. In this research, we adhere to the definitions proposed by [43] that classifies the approaches to Edge AI as layers of a pyramid. Starting from the Cloud scenario, each subsequent level moves a part of the process towards the edges, having in the highest tier (Level 6) the whole process performed within the end devices. To be precise, with device it is identified any potential device that presents the properties of being of users' propriety and usage. Thus, in this category, falls both personal and company-issued devices such as laptops and smartphones, but also IoT devices such as routers, cameras, sensors.

As described in this section, Fig. 1 will describe and discuss the properties and differences of these six approaches (compared to the well-known cloud approach).

An entirely different subject is the application of AI to optimise and self-configure the network slices that provide the services, a topic defined as *Intelligence for Edge* or *AI for Edge* [12, 33]. In the same surveys, the authors reported a collection of technologies designed to work at the edge of the networks; we suggest the readers refer to them, alongside with the numerous surveys in the area [11, 18, 19, 23, 26, 32], for detailed information regarding the subject.

The SECaaS theory provides an example of this semantic difference. Generally speaking, a network IDS, such as the DGA-based botnet detection framework in-here presented, might be configured as a service using any virtualisation technology. Within the 5G ecosystem, this IDS would be designed as virtual network function (VNF) to be deployed as other services by the orchestrator. Similarly to 5G orchestration-level intelligence, AI for Edge [12] encompasses those ML applications that provide optimisation and learning capabilities to the management modules. On the contrary, AI on Edge covers the study of what kind of intelligence should be deployed in the IDS service, not how to deploy it; *e.g.*, how to separate malicious AGDs from legitimate FQDNs.

As for 5G orchestration, the location of the deployment of the service does influence the performances, especially in the context of the depicted use case of DGA-based botnet SECaaS. Thus, to highlight the differences between the approaches, in Fig. 1 are reported the prototypical architectures for each level in regards to the training (indicated with a red diamond marked with the TR acronym) and inference (indicated with a blue triangle with the INF acronym) phases. In this novel computation paradigm, low requirements tasks can be executed at the edges of the network, often directly on the end devices.

On the one hand, classical approaches rely upon the cloud datacenters for training the AI models [33], *i.e.*, the most resource-consuming part of the process. Whether it is possible to perform the training phase on the edges—and ultimately on the end-devices—heavily depends, among others, on:

(i) the use case, as not all scenarios have critical issues that can be mitigated by offloading the intelligence to the edges;
(ii) the latency and delay requirements, as, depending on the use case, there might be some constraints on the privacy requirements, *e.g.*, personalised user experience that strictly requires that no user data leave the device [16];
(iii) the isolation requirements as coexisting applications, cryptographical restrictions, and network slicing limitations may cause privacy and data issues;
(iv) the amount of data to be processed, *e.g.*, a face recognition service that needs to process hours upon hours of multiple feeds necessitate a non-trivial amount of bandwidths;
(v) the resource constraints of the end-devices, especially in the Internet of Things (IoT) ecosystem where the issues related to battery usage, broadband connection unreliability are more noticeable.

On the other hand, generally speaking, the data originate at the most remote edge of the network, *i.e.*, in the end-devices. As the data needs to be processed and inferred by the AI models, it is reasonable to aim at completing as much computation as possible directly on the devices (Levels 3 and 6 of Fig. 1) or eventually offloaded to the edge (Levels 2, 4, and 5). Nonetheless, moving ML components to the device has some limitations, mainly regarding limited energy, computing capabilities, and storage [13].

Finally, and referring to Fig. 1, it is worth mentioning that only two levels, namely the third and the sixth, guarantee that the users' data never leaves the device. In Fig. 1, such limit is indicated with a double line, that can be either

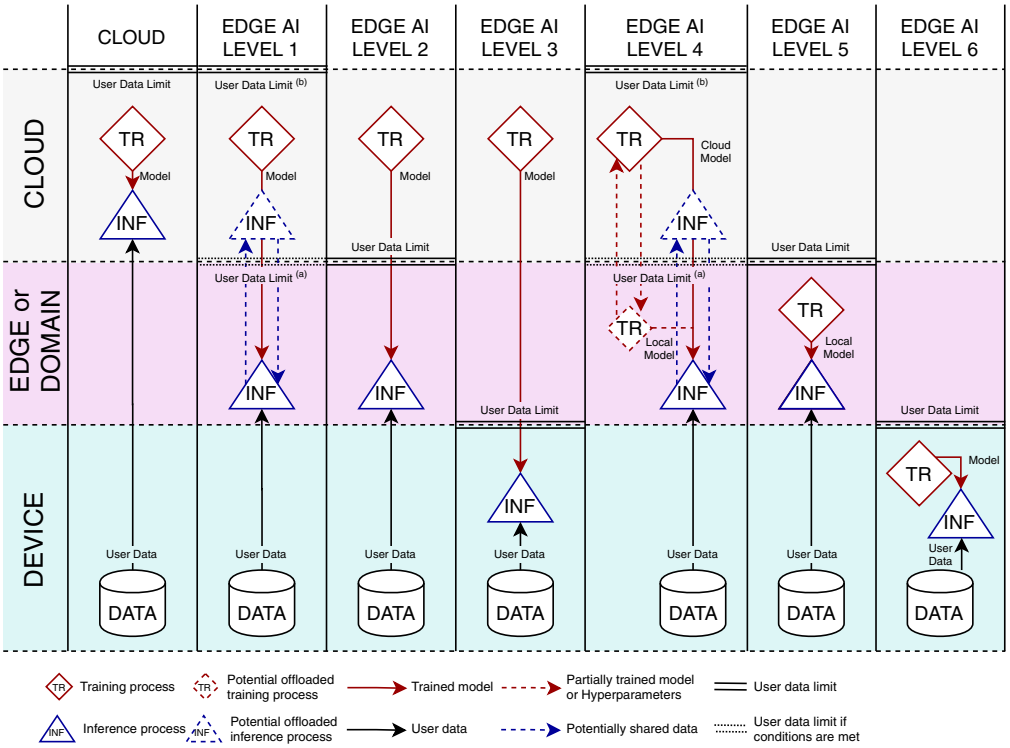


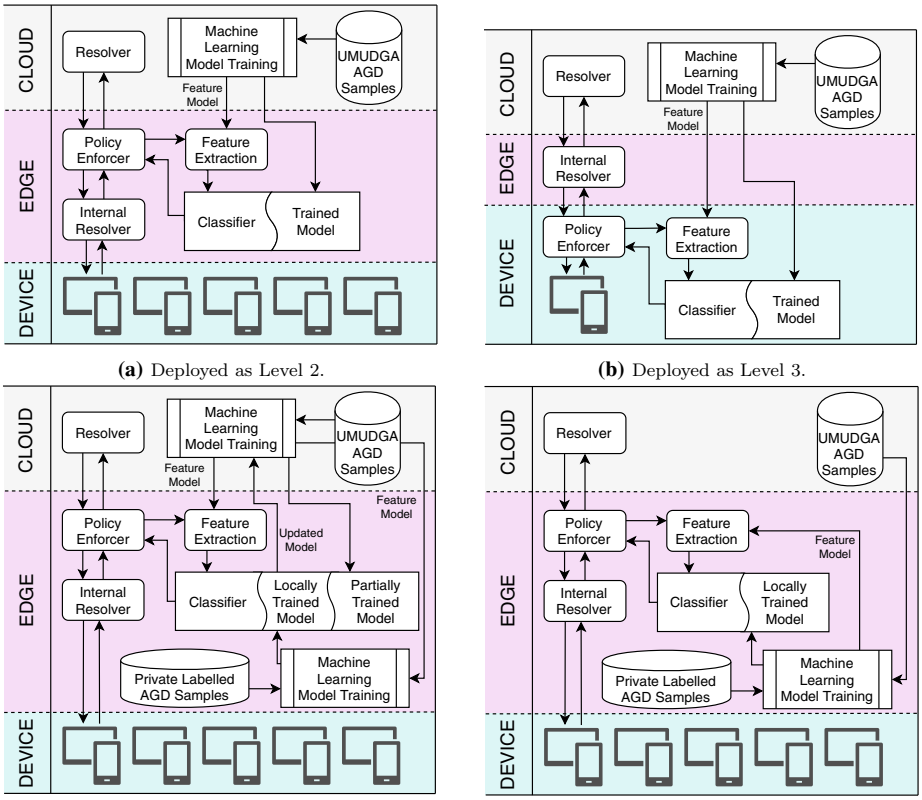
Fig. 1 Comparison overview for Edge AI protypical architectures. [a/b]User data limit when the inference process is carried out at the edge level (**a**) or at the cloud level (**b**)

continuous or dotted. As will be described later on, in this last case, it represents the actual data limit in the case where no cloud-offloading is activated. Similarly, in the second and fifth levels, the inference process is performed on edge. Since these services are deployed closer to the users, there is a significant reduction in both latency and bandwidth; however, the security implications of leaving the data at the edge level are still unclear [18, 22, 23]. To be precise, if multiple edges merge their data to the cloud, privacy issues might arise, *i.e.*, the privacy concerns gathering around the extensive use of AI on end-devices are not related to the devices themselves, but with how and where the data are transmitted, processed and stored. Nonetheless, one of the main advantages of the centralised cloud relies on this uploaded and shared knowledge. In a collaborative environment such as has been studied before [27], IDSs can benefit from the intelligence gathered from multiple sources. Specifically, collaborative, cloud-based IDSs have been proved effective against zero day (0-day) malwares [8].

2.2 Architectural differences

With regards to the different options presented in Fig. 1, this section will introduce a brief analysis of the most engaging aspects. To do so, we will introduce the Fig. 2 that presents a vertical analysis for Levels 2, 3, 4, and 5. To be precise, this section will unfold the options and capabilities of a DGA-based detection framework to be deployed as a collection of SECaaSs.

To start with the inference process location, Fig. 2a, b present two edges configurations. To begin with, Fig. 2a reports a prototypical architecture for a 5G-like network slice that interposes between the internal resolver and the remote ones. In this scenario, the network slice is configured to apply a policy enforcer whose rules are defined by, for example, a ML classifier. Similarly, Fig. 2b reports a prototypical architecture for a on-device resolver that intercepts user's requests before sending them. In both configurations, the requested FQDN is extracted from the DNS query, processed by the feature extraction



(a) Deployed as Level 2.

(b) Deployed as Level 3.

(c) Deployed as Level 4.

(d) Deployed as Level 5.

Fig. 2 Architecture for a DGA-based botnet detection framework deployed at different levels of Edge AI framework

microservice, and classified using a pre-trained-cloud model. The difference between Fig. 2a, b dwells in the actual location of this process, *i.e.*, the edge and the end-device. Notice that both configurations require a trained model, which has been conveniently trained in the cloud. Mind also that there are no obligations to share the inferred data with the cloud services; thus, both the isolation and the privacy requirements can be achieved.

Similarly, Level 4 (Fig. 2c) and Level 5 (Fig. 2d) achieve the same isolation property by analysing the data directly on edge, without offloading it to the cloud. To better discuss it, it is necessary to refer back to Fig. 1, and specifically Levels 1 and 4. In both cases, the inference process takes place on edge; however, the architecture does leave open the possibility of offloading such tasks to the cloud services if needed. To put it in other terms, since the edges are generally resource-constrained, they might decide to offload part of the inference to the cloud services to optimise the workload. Offloading strategies include partial offloading, vertical collaboration, and horizontal collaboration, to cite a few (the readers might refer to [33] for a detailed survey on the subject). The main difference between the two approaches resides in the possibilities offered by transferring part of the training to the edges. That is to say, in Level 4, each edge can update the model and eventually share it with the cloud in a federated fashion. Even though the fourth level could be, in theory, designed to transfer the data to the cloud for load balancing purposes, in the scenario depicted in Fig. 2c, this is not the case. The classifier is designed to work only on edge; thus, the data limit can be considered on the edge boundaries, and not on the cloud ones.

So far, both Fig. 2a–c have the training phase deployed as a cloud SECaaS. By contrast, in Fig. 2d the same process is configured to have the training phase on edge. To be precise, in Fig. 2d the in-edge training process uses a cloud-collected dataset that is augmented with private and locally-available data. It is worth mentioning that the

training process does not require both local and cloud samples. However, shared knowledge [22, 27] is effective in tackling cybersecurity threats, specifically in the case of 0-days malwares. Although the scenario described in Fig. 2d offers a collaborative base for sharing knowledge, it does not include the actual share of the trained model (or its hyperparameters) as Fig. 2c (Level 4) does. In Fig. 2c, we assume that the edges are cooperating to produce an optimal model without sharing users' data, thus enforcing the isolation and privacy requirements. On the contrary, in Fig. 2d, the edges cannot share the user data to the cloud, hence forcing the central cloud to obtain its data somewhere else. In any case, a hybrid architecture is achievable, where multiple edges adopt a different model depending on local requirements: for example, in a context such the one offered by the DGA-based botnet detection, sharing users' browsing history might not be feasible without substantial anonymisation. Nonetheless, as proved in [37, 39], sharing a list of domain names (both legitimate FQDNs and AGDs) does not constitute a privacy issue.

2.2.1 Outlined differences, pros and cons

To summarise the different approaches, Table 1 reports their properties, advantages and disadvantages when the architectural models are applied to the DGA-based botnet detection.

For this comparison, it is worth mentioning that the difference between a 5G-style edge environment and a corporate domain one is of utmost importance, being it key to understanding the potential risks and benefits associated with cloud data [29]. In this scenario, several assumptions are, in fact, reasonable. Besides small and micro-enterprises, it is plausible to consider a private cloud environment, perhaps with a dedicated IT team or even a security operation center (SOC) in case of larger corporations. The resources available to these entities are not the same as those at the disposal of 5G-RAN nodes, albeit they share some advantages and disadvantages of Edge AI. Firstly, from the computation resources available, cloud training and testing feature virtually infinite resources; however, farthest edge pieces of equipment have limited means while end-devices have substantial constraints. Not surprisingly, latency heavily depends on the proximity to the users, the closest to them, the fastest the response. The

Table 1 Qualitative comparison of different architectural approaches for a DGA-based detection framework compatible with Edge AI

Metric		DGA-based SECaaS deployment location			
		Cloud	Corporate domain private cloud	Edge	End device
Computational resources availability		Unlimited	Depends on corporate resources	Limited	Constrained by power, battery and performance
Latency		Potentially high	Significantly high	Low	Zero
Data source availability	*Labelled*	Significantly high if in a collaborative framework	Depends on the size, scope, and availability of SOC	None	None
	Not labelled	Potentially unlimited if in a collaborative framework	Limited to corporate nodes	Limited to node	Limited to device
Data access and isolation		Third party accessible and variable with ToS	Corporate	Service provider	User
Benefits from shared intelligence	*Receiving data or models*	Multiple data sources, early identification of new threats	Broad Protection		
	Sharing data or models	Higher protection for the framework participants	Might get economical benefit from sharing data or models	Directly none. Indirectly, it contributes to the ecosystem	
Risks from shared intelligence	*Receiving data or models*	Data or model poisoning			
	Sharing data or models	Security CIA non-compliance, potentially higher risk of exposure due to higher data value			Security CIA non-compliance

concept of data source availability is deeply connected to the sharing framework between the participants, in this context, such as presented by [22], the cloud can either receive aggregated and anonymised data or a selection of parameters to update accordingly, ultimately providing a virtually unlimited overview of the network. The economic resources available at the highest levels also justifies the assumption of high-quality labelled data originating from the ones shared by the edges. The same happens within medium to large corporations, where a dedicated SOC might be providing curated data for the AI processes to work. For the same reason, data access can be limited to corporation assets, thus providing a similar isolation level as edge devices; as widely stated before in literature, once the data reaches the cloud, there is no guarantee besides what the service provider declared in the term of service (ToS).

In general, the strength of deploying a DGA-based botnet detection on edge relies on providing broad protection services (due to the collaborative efforts [27]) closer to the user, provided that the security triad (Confidentiality, Integrity, and Availability) of the data is guaranteed. However, as for every other collaborative framework, defensive mechanisms against data and model poisoning must be considered [34].

Finally, in terms of computational requirements for Edge AI, Khan et al. [19] reported a curated syllabus on expected technologies that are, in general, required for enabling Edge AI, while Coppolino et al. [11] specifically discussed hardware-based enhancement to make Edge AI feasible. Furthermore, Lin et al. [22] presented a comprehensive survey on actual implementations approaches, architectures and libraries to enable Edge AI in a decentralised and collaborative fashion.

2.2.2 Critical aspects of services location

To summarize, cloud services provide, by far, the most common scenario. In this context, a broadly employed—and thus lacking in innovation—approach is to propose frameworks that can provide SECaaS through self-deployable VNF [2, 40]. For these functionalities, most authors across the board agree on primary key performance indicators (KPIs) such as detection rates, yield capabilities, and guaranteed availability, indeed, the cloud scenario is characterised by scalable and high-reliable VNF that provide the required services to massive amounts of customers. Amid the criticalities, there are the privacy issues, well discussed in the past [20], that arise with sharing the data to third parties (first and foremost the GDPR-related issues). Nevertheless, aggregating and collecting a massive amount of data can provide useful information to CERT and nation-wise monitoring actors without harming users' privacy.

With regards to DGA-botnet detection, the scenario itself does not offer any relevant challenge besides the ones already identified and explored in literature. Although supported by practically unlimited resources, SECaaS at cloud level suffers from scalability issues in regards to the number of connected devices, a critical aspect for B5G ecosystems.

Similarly to the cloud services, a few authors have designed SECaaS bearing in mind the potentialities of edge computing, for example, by pushing the VNFs to the remotest areas of the networks. Regarding the KPIs, and besides those already defined for the cloud services, authors have collected and defined several metrics based on the hardware requirements needed for running the VNF. Although some authors make the distinction between the different categories of KPI indicators (*e.g.*, quality of service (QoS) and quality of experience (QoE) among others), such a discussion falls outside of the scope of this research. Indeed, this research field hankers for a precise and formal definition of metrics and indicators to enable a quantitative comparison between reproducible frameworks. To cite an example of such indicators, there is the capacity of utilisation, *i.e.*, the percentage of classification capacity used over a predefined time unit that enables optimised load balancing for network intrusion detection [15].

To further discuss the applicability of a DGA-based detection module at the edge level, it is necessary to separate the concept of edge and corporate domain.

Edge computing—On the one hand, the edge-related discussion should pivot on the automated, human-free, capabilities, highlighting themes such as the limited computational resources, the extremely low-latency requirements and the native isolation from the cloud services.

Corporate domain—On the other hand, corporate domains can be partially overlapped with the concept of the private cloud, thus centring the discussion on themes such as the protection and isolation of the sensible data rather than fully automated detection capabilities. The private cloud environment provides mitigation against the already mentioned privacy issues of the cloud providers, without tackling the benefits of the broad protection that a shared knowledge base can provide [9].

Provided that the corporate domain can guarantee the security CIA compliance, and as previously reported in Table 1, the scenario can present several benefits to the SECaaS, specifically in terms of the amount of data and overall network visibility. In comparison, a fully automated on-edge configuration would permit much faster responses due to the reduced latency and high self-capabilities. However, a dedicated SOC, if available, can be handy in

detecting anomalies and new APTs, thus providing constant updates and improvements to the detection models.

Lastly, only a few authors have proposed detection solutions that work entirely on the end-devices [14, 23]. In this constrained scenario, hardware related KPIs becomes critical, *e.g.*, execution time and resource consumption. Bear in mind that the end-device category does not only include smartphones and personal devices, but it also encompasses a broad category of IoT devices such as home routers and Industrial Internet of Thingss (IIoTs) sensors. Indeed, Edge AI solutions proliferate in industrial scenarios due to the low resources, high availability constraints [35].

2.3 Edge AI and cybersecurity

There are a few notable solutions worth mentioning concerning the cybersecurity side of the applications compatible with the Edge AI paradigm.

Besides the inherited threat model from the cloud, edge computing introduces new security risks due to its traits [23]. Ideally, a fully privacy-preserving approach would only manage strictly necessary data without transmitting them to any external processing centre (in Fig. 1, only levels three and six achieves this status). We invite the readers to refer to [12, 22, 33] for technical analysis of up-to-date libraries to deploy frameworks in such fashion.

On the attack side of the cybersecurity, Isakov et al. reported an in-depth analysis of the current state of the art regarding the threats [18]. In their survey, they firstly introduced a taxonomy on vulnerabilities and criticalities of neural networks that can be exploited by criminals to gain an advantage on the cyber defences; also Liu et al. [23] reported a collection of security threats that target data explicitly.

On the defensive side, several aspects need to be taken into account. First and foremost, most (if not all) data have protection and isolation requirements; consequently, the security challenges associated with those subjects are of the highest importance and priority. In such a sense, several surveys have collected, analysed, and compared specific mitigation techniques to the vulnerabilities that have been gradually uncovered. Among them, Liu et al. [23] analysed the phases of collection, processing, and storage of the data management on the edges, with particular attention to the open challenges and future research directions.

For another aspect of the defensive side, classical ML powered detection frameworks are evolving toward decentralised, edge-oriented solutions. Two scenarios prevail among others when looking at the research interest; on the one hand, the industrial environment offers requirements for high-availability, low-resources and low-latency services. The derived research challenges cause the research community to thrive [35, 41]. On the other hand, the challenges and requirements offered by the 5G self-protection scenarios mainly lead the community to design IDSs as full-cloud services [24], having just some of them offloading part of the inference process to the edges [15, 17]. In that sense, [14] stands out by presenting a working solution for a Level 6 (fully on-device) architecture for IoT devices. Notably, mixed solutions like the ones proposed by the federated learning paradigm [22, 32], *i.e.*, those solutions where the training is at least partially performed on edge are hyped and are notably worth further researches.

Within network detection, to the best of our knowledge, this is the first attempt to explore DGA detection on the edges. Notably, several frameworks have been proposed just in the last year [3, 28, 31, 36], however, besides apparent reproducibility issues [39], every work allegedly resolves the challenges related to DGA-based botnet detection. Nevertheless, there is a clear trend in terms of the chosen technology, *i.e.*, Long Short-Term Memory network (LSTM)-based and, in general, DL-powered framework [28, 31, 36].

In addition to generic network-based malware detection, a particular focus should be dedicated to the APT threat [21]. These advanced malwares are often recompiled for the specific target and present multiple obfuscated variants that do not match classical signatures. However, the extensive usage of machine learning in log and data mining has been proved useful to detect malware infection symptoms, especially new ones (0-day). In the context of this manuscript, the same ideas are applied to the network communication phase [39] to identify a botnet communication as early as its first connection to the Command & Control (C&C). In the context of Edge AI and, in general B5G, it appears that this threat explodes with the number of newly connected and poorly protected devices [35].

Last but not least, the distributed nature of these architectures requires to discuss the underling security primitives that guarantee the confidentiality and integrity of the shared data [29]. However, subjects like trust and reputation management alongside with access control, authentication and encryption will not be discussed in this research article as they are out of scope and they have already been widely covered in literature [29].

3 Experimental DGA-based botnet detection on Edge

Most of the Edge AI reported in the previous sections offer little-to-no architectural challenges besides the one offered by implementing the actual algorithms on the resource-constrained edge devices. Despite that, the

Federated Learning approach offers an intermediate solution that connects the best cloud performances with the isolation and scalability offered by the edge characteristics. The remaining of this section is structured as follows: Sect. 3.1 depicts the framework's architecture, while Sect. 3.2 presents a brief remark in terms of minimum computational resources and modules locations.

3.1 Proposed architecture

The framework described in the paper at hand is designed to be compatible with the Edge AI Level 4 in a Federated Learning fashion. Figure 3 presents its architecture, including both the cloud level services and two different edge-level environments leveraging the idea of a collaborative framework The edges, represented as isolated domains, receive a shared and pre-trained detection model that can be augmented with locally available data. However, albeit it is possible to improve the model in such a way using the same data structure (*e.g.*, feature set and format, among others) it is also true the opposite. Locally poisoned data could skew the model in an adversarial fashion [30, 34]. Similarly, each domain can willingly provide data or partial models to be included in the shared cloud next training iteration. Bear in mind that, as will be demonstrated by the experimental results, the whole edge plan might be shifted to an end-device, provided that the libraries for executing the code are available. To the best of our knowledge, this is the first attempt to establish a DGA-based botnet detection framework that can be deployed this far in the edges.

At a high level, Fig. 3 presents three areas, namely the Cloud level (on the top half, with a white background) and two edge levels (on the bottom half, with yellow and blue backgrounds). Edges are also identified as "domains" due to the extendibility of the proposed framework to an enterprise scenario. In such a case, the company infrastructure might be composed of several edges that rely on a single shared training subcomponent.

One of the crucial advantages of having enforced the separation of the detection modules relies on the data and model isolation [22]. The cloud components can be seen as multiple SECaaS provided by a cybersecurity vendor, while each edge/domain represents a federated subscriber. In Fig. 3, "Domain A" does not share model nor data to the cloud, simulating a restricted environment where the collected user data cannot be pushed to the cloud. No information regarding eventual APTs and 0-days are thus available for the federation. On the contrary, "Domain B" does share the pieces of evidence and samples of collected malwares, enabling the cloud model to be updated with the new information. Instead of data, the hyperparameters of the model could be shared to provide an extra isolation level [22].

Among the various levels identified in Fig. 1 and discussed in Sect. 2, we considered Level 4 as the one that provides both the flexibility of deployment required by automated 5G scenarios and the compatibility with the isolation requirements characteristics of corporate environments. To be precise, as will be discussed in Sect. 3.2, the whole process could be executed directly on the end-devices (thus achieving the Level 6 compatibility);

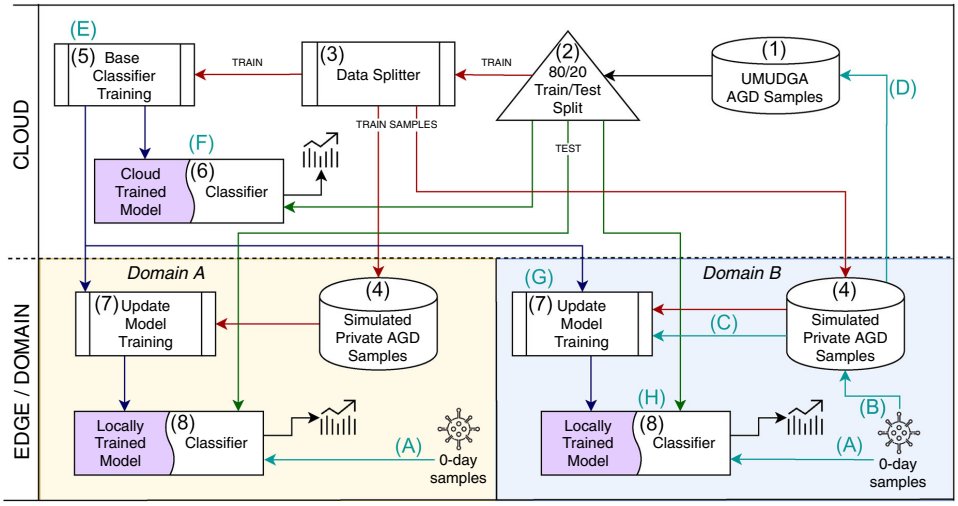


Fig. 3 Architecture and data flow for the proposed L4-compatible experiment

however, we deemed more interesting to evaluate the potential flexibility offered by the federated learning environment, *i.e.*, a collaborative network of virtualised and lightweight SECaaS to provide DGA-based botnet detection at scale.

In this configuration, the two domains represent the duality of the Level 4 architecture design presented in Fig. 1. The user data limit is, indeed, different. In fact, for domain "Domain A", that does not share user data with the central cloud, the limit is enforced at the boundaries of the edge. Independently from the location of the classification process, "Domain B" does share data with the cloud provider, thus moving the limit mentioned above to the boundaries of the cloud services. To summarise, the isolation properties heavily depend on the actual deployment configuration and application scenarios; policies and restrictions might apply to different use cases, as well as risks and benefits.

In terms of the chosen technology, besides several DL algorithms that can be used for DGA-based botnet detection [28, 31, 36, 37], we will adhere to the explainable AI philosophy in the current manuscript. Indeed, although DL solutions are trending, this research will empirically demonstrate that classical and lightweight models can achieve excellent results. Albeit Federated Learning focuses on DL, the same principles can also apply to some classical models, first and foremost the tree-based ones. As a consequence, all the experiments will be carried out with decision-tree based solutions like the Random Forest algorithm or modern approaches such as the XGBoost or LightGBM.

The remainder of this section has been divided into three parts to ease the framework exploration process; firstly, the data are described and presented in Sect. 3.1.1; then, the primary framework's loop (*i.e.*, steps from (1) to (8)) is analysed in Sect. 3.1.2; finally, an example of the feedback loop (*i.e.*, steps from (A) to (H)) is provided in Sect. 3.1.3 in the advent of a new 0-day infection.

3.1.1 Preprocessing and feature analysis

The proposed experiment uses the data collected by the authors [39] and publicly released in [38]. In the dataset, 50 malware variants have been collected and described, providing both the raw lists of AGDs and a preview of the extracted features. Nevertheless, not all reported features provide enough information to be used for detection purposes [37, 39]. Hence, a feature selection process has been carried out to limit the overheat provided by the curse of dimensionality, also given the context of low-resource or resource-constrained devices provided by the Edge AI scenario [37]. Furthermore, some malware classes have been grouped due to their indistinguishability [39] with a combination of clustering techniques and careful human revision. Of the 50 malware families identified, 21 clusters have been identified. Figure 4 reports the classes with the associated clusters. The clustered data have been resampled (200,000 FQDNs for the training set, and 9628 FQDNs for the testing set, both stratified).

Features are ranked with a recursive feature elimination process, eliminating one feature per iteration. Experimental results suggest that the top 10 features are representative enough to achieve good classification results. Table 2 reports these classification performances for the three classifiers picked for the analysis. By combining the information reported in Table 2 with the one reported in Table 3 it is possible to notice that the average resource consumption halves by accepting a 2% loss in F1 score. The proposed trade-off enables scalable and dynamic reconfiguration of the detection model, but simply switching to the most reactive and less computationally expensive model depending on the traffic volumes. To be precise, Table 3 reports the results in terms of resource consumption's for both the full feature set and the top 10 feature sets.

3.1.2 Data flow and experimental results

For the experiments, the framework samples 10,000 domain names for each provided class obtained from [38], and the framework is built upon the Random Forest classifier with the warm start option enabled to allow the in-edge upgrade of the model.

The data flows from a central shared data source, indicated in Fig. 3 with the number (1) to the edge classifiers indicated with the number (8). The elements marked with circled letters are instead analysed later in Sect. 3.1.3.

A standard 80/20 separation is adopted for splitting the data in step (2). On the one hand, the resulting testing set (20%) will be used at cloud and edge levels to ensure comparability results. On the other hand, the training data (80%) is shuffled and separated into three different, possibly stratified sets with configurable proportions in (3). The subcomponent separates the data to simulate data that are available only at the edges level (4).

The cloud training module (5) will use the first data set to train the base model (6) to be shared with the edges. The first experimental results are gathered at this phase by evaluating this classifier model against the testing set and reported in Table 4.

The base model is then shared with all edges belonging to the federation and augmented with locally collected data (7); these data are simulated by sharing one training set with the edge in step (3). The updated model is evaluated in (8) against the same testing data used in the previous phase; the results are also available in Table 4. From the

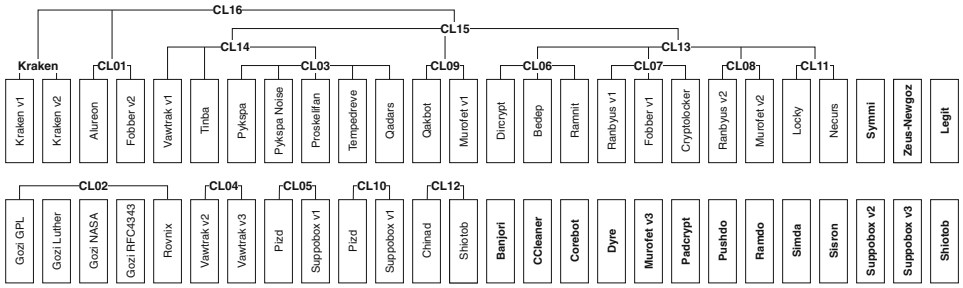


Fig. 4 Partial Hierarchical Cluster of UMUDGA [38] classes, all the non-connected boxes are considered standalone classes

Table 2 Model performances comparison for feature selection

Features	Algorithm	F1 micro	F1 macro
Full	Random forest	0.9523	0.9514
Top 10	Random forest	0.9399	0.9386
Full	LightGBM	0.9612	0.9604
Top 10	LightGBM	0.9473	0.9459
Full	XGBoost	0.9024	0.8988
Top 10	XGBoost	0.8794	0.8749

table perspective, it is possible to notice that the results provided by updating the model (7) and validating it against the testing set (8) do not necessarily lead to an improvement in the model. In other words, retraining the model with additional data might not result in an increased detection ratio [30]. Nevertheless, this local-update functionality is of undoubted interest, and future researches might consider it when exploring security architectures.

3.1.3 Simulating a zero-day APT

One of the critical aspects of having ML-powered detection modules is the capability of identifying suspicious behaviours never seen before. This experiment aims to simulate the framework data flow in case of a new malware (which could be easily interpreted as an early stage APT infection).

While referring to Fig. 3, this experiment flow is identified by circled letters (*i.e.*, from (A) to (H)) instead of numbers. In step (A), a few AGD samples are injected in the analysed set of data to simulate a previously unseen malware establishing a communication channel with the C&C.

Two scenarios are available at this point, *i.e.*, depending on the Edge concept's interpretation, some capabilities might or might not be enabled. On the one hand, in "Domain A", (that simulates a classic, fully automated, 5G-RAN environment) the new threat will remain unnoticed. On the other hand, in "Domain B" (that simulates a corporate domain with active monitoring tools), the SOC eventually notices the confidence decrease (see Table 4, row (A) Edge Test Zero Day) and investigate the matter. In

Table 3 Model resources consumption comparison for feature selection (duration are in seconds, memory usage is in MiB)

Feat.	Algorithm	Type	Total time	Instance time	Memory peak	Memory increment
Full	RF	Train	75.243	0.00038	1157.18	395.82
Full	RF	Test	0.630	0.00007	1066.05	12.23
Top 10	RF	Train	31.555	0.00016	1115.76	363.69
Top 10	RF	Test	0.651	0.00007	1124.62	12.27
Full	LightGBM	Train	305.845	0.00153	1377.81	641.18
Full	LightGBM	Test	2.436	0.00025	757.02	0.34
Top 10	LightGBM	Train	45.275	0.00023	768.75	9.84
Top 10	LightGBM	Test	1.627	0.00017	769.01	0.26
Full	XGBoost	Train	1669.997	0.00835	1543.91	595.18
Full	XGBoost	Test	1.529	0.00016	1345.97	0.31
Top 10	XGBoost	Train	261.434	0.00131	757.41	2.48
Top 10	XGBoost	Test	1.411	0.00015	757.49	0.08

Table 4 Random Forest performances for the simulated experiment, before and after injecting the zero day

Phase	Trees	Classes	Samples	0-day	F1 micro	F1 macro	Confidence
(5) Cloud training	50	21	127,761	✗	0.929	0.928	0.957
(6) Cloud testing	50	21	39,926	✗	0.931	0.930	0.910
(7) Edge retrain	50 + 10	21	159,702	✗	0.931	0.930	0.935
(8) Edge test	50 + 10	21	39,926	✗	0.931	0.929	0.906
(A) Edge test zero day	50 + 10	21	8000	✓	0.776	0.855	0.812
(E) Cloud retrain with Zero Day	50	22	128,761	✓	0.930	0.931	0.957
(F) Cloud testing	50 + 10	22	47,926	✓	0.943	0.933	0.925
(G) Edge retrain	50 + 10	22	187,725	✓	0.929	0.929	0.934
(H) Edge testing	50 + 10	22	47,927	✓	0.943	0.937	0.923

this last scenario, some samples will be eventually collected (B) and used to update the local model (C). Possibly, and in a federated learning fashion, these data could be shared and aggregated at the cloud level (D), to be studied ((E) and (F)), and eventually distributed to some edges (G) as the base model. Similarly to the forward-loop (steps (1)-(8)), the results are reported in Table 4.

3.2 Resources requirements

A computation analysis has been carried out to evaluate the various components capabilities and requirements. The results, presented in Table 5, encompass both the time (Table 5a) and the memory (Table 5b) requirements of the subcomponents. In the table, the training process indicated in Fig. 3 as (5) has been divided into the initial generic fitting and the optional cross-validation process. Similarly, the clients have been divided accordingly to the option availability of a retraining component (indicated in Fig. 3 as (7)).

To begin with the experiment configuration, a unique VNF has been configured to execute each submodule independently. On the subject of resources allocated, several typical profiles have been taken into account, ranging from 1 dedicated core and 1 GB SDRAM (typical configuration of a Raspberry PI IoT device) to 4 dedicated cores and 4 GB SDRAM (typical configuration of a low-end personal laptop). With the specific configuration of the chosen ML model, lowering the minimum memory resources is not possible. Nevertheless, new specialised IoT-oriented ML libraries might provide comparable detection performances while operating under lower resources requirements. Both the results reported in Table 5 and previously in Table 3 demonstrate that even without optimised mobile-specific ML libraries it is possible to achieve excellent and explainable detection results in a resource-constrained environment.

4 Lessons learned and future works

Edge AI represents an innovative solution to several high-ends cybersecurity issues. While local models excel at keeping the information isolated from others, they also require a non-negligible amount of shared data to be able to target threats effectively.

In other words, while predictive models such as [16] works well by only using the individual user data, other detection models–such as the one proposed in this research–are not suitable to work individually. To be precise, detection models benefit from the shared knowledge base gathered either at cloud level or collaboratively in an edge-federated fashion; local models are great to learn the particular context in which they are deployed, but they miss the broad vision that only the cloud can provide. Indeed, some scenarios might require extensive computations for preprocessing before the actual inference process kicks-in; for example, in the context of IIoT, most of the sensors deployed do not have the minimum resources (including battery capacity) to perform any intensive computation [35]. For the DGA-based botnet detection, tackling the malwares at the DNS level enables to deploy lightweight platform-independent probes capable of privacy-aware real-time inspections at scale. Future researches should include the tradeoff between performances, latency, and privacy aspects in the architectural design.

Concerning the data protection subject, future researches might focus on ways to aggregate and anonymise data (*e.g.*, homomorphic and searchable encryption), knowledge transfer learning, gossip training, as well as explainable ML (and DL) models which hyperparameters could be shared within the collaborative network. The parameter sharing approach could potentially remove a substantial amount of computation and still provide a powerful detection suite. In this sense, future works might include detectors capable of preemptively block connections that

Table 5 Performances of the machine learning processes depending on the resources dedicated to the virtualised environment

Role	Component	Time (s)									Med.	STD
(a) Time requirements												
Server	(5) Fitting	65.05	60.55	79.05	64.70	76.26	64.04	64.03	63.81	63.56	64.04	6.33
	(5) 10 CV	236.30	235.92	274.03	13.20	14.35	13.44	15.02	13.85	14.49	14.49	117.86
	(6) Validation	0.31	0.31	0.35	0.31	0.35	0.32	0.70	0.33	0.33	0.33	0.13
Client w/ retrain	Load model	0.10	0.10	0.15	0.09	0.14	0.12	0.12	0.15	0.10	0.12	0.02
	(7) Add. fitting	2.73	2.78	3.01	2.80	3.07	2.91	2.94	2.90	2.93	2.91	0.11
	(8) Validation	0.37	0.36	0.41	0.45	0.41	0.37	0.46	0.48	0.47	0.41	0.05
Client w/o retrain	Load model	0.08	0.11	0.11	0.12	0.13	0.13	0.14	0.13	0.10	0.12	0.02
	(8) Validation	0.32	0.31	0.35	0.34	0.37	0.34	0.34	0.31	0.32	0.34	0.02
	Processors	1	1	1	2	2	2	4	4	4		
	Memory	1 GB	2 GB	4 GB	1 GB	2 GB	4 GB	1 GB	2 GB	4 GB		

Role	Component	Memory usage (MB)									Med.	STD
(b) Memory requirements												
Server	(5) Fitting	251	251	251	266	266	267	294	294	299	266	20
	(5) 10 CV	677	677	677	806	827	825	764	917	970	806	106
	(6) Validation	427	426	426	540	561	558	470	624	671	540	91
Client w/retrain	Load model	410	410	410	409	410	410	409	410	410	410	0
	(7) Add. fitting	410	410	410	427	424	427	435	432	434	427	11
	(8) Validation	410	410	410	432	430	431	442	437	442	431	14
Client w/o retrain	Load model	410	410	410	410	409	410	410	410	410	410	0
	(8) Validation	410	410	410	410	410	411	417	417	418	410	4
	Processors	1	1	1	2	2	2	4	4	4		
	Memory	1 GB	2 GB	4 GB	1 GB	2 GB	4 GB	1 GB	2 GB	4 GB		

are suspected of belonging to botnet networks by merely analysing the DNS queries.

On the subject of traditional ML vs DL solutions, we deem that a remark is needed: DL is not required in every domain. In the depicted scenario, the DGA data does not feature complex non-linear relationships. As established before [37, 39] and remarked in Sect. 3, a small number of features will suffice to train classical ML algorithms achieving good results. As empirically demonstrated in the section, a fully on-device DGA-based botnet detection is possible with traditional ML algorithms, given that enough samples for each class are provided. Moreover, most approaches focus on supervised learning as it provides relatively straightforward and verifiable pipelines; despite their clear benefit, supervised solutions require large datasets and careful data supervision. Unsupervised approaches have also been explored, although not in-depth [37], and future researches might focus on analysing semi-supervised hybrid solutions to take advantage of the massive amounts of unlabelled data.

For another key lesson, attacks at collaborative models are not something new [30], as a such, DGAs are going to evolve to mimick legitimate FQDNs and potentially to tackle the detection algorithms directly in an adversarial fashion. On-edge solutions, and precisely in a federated learning ecosystem, have been proved susceptible to adversarial attacks [22]. In such a sense, future works might discuss whether edge computing is necessary, desired, or even feasible.

Finally, as previously indicated by the authors [37, 39], the field in-here studied features a general lack of reproducibility:

- First and foremost, the data used to power the frameworks are rarely shared; therefore, future researches should make use of already shared and well-known datasets, or provide their own with the appropriate comparisons to the state-of-art.
- Secondly, the deployed models are described, but not released (often neglecting to comment on the hyperparameters configuration); as such, future researches

should focus on producing reproducible results that can be tested and validated by the community.
- Thirdly, although quantitative comparison frameworks for ML (and DL) algorithms do exist, their application is often limited to aggregated indexes that might deceive the results, *e.g.*, the Area Under the Curve (AUC); hence, future researchers should report all the outcomes, especially the negative ones.
- Lastly, the actual implementations are often described and tested outside a proper validation framework; in such a sense, future researches might focus on developing a suite of tools and indicators to enable formal comparisons.

5 Conclusions

Albeit the disruptive innovation led by the 5G enabling technologies, researchers did not stop in exploring more advanced solutions. Among others, Edge AI is believed to be the next enabling technology for what is coming beyond the 5G. It is in this direction that collaborative concepts, such as federated learning, empowers the raw potential of 5G bandwidth and number of devices, combining it with lightweight but highly effective machine learning models. On-device AI might, in some cases, represent the perfect solution for data-sensible scenarios; however, Edge AI aspires to serve as an intermediate compromise between cloud practically unlimited resources and privacy constraints. With this scenario in mind, this research empirically proves that a DGA-based botnet detection module is not only feasible to be deployed in the resources-constrained environment, but it also would benefit from shared intelligence in a federated fashion. A natural evolution of this research would be to extend the detection models also to tackle frequent domain name attacks such as the typosquatting (*i.e.*, URL hijacking) or to identify less-known techniques such as the DNS-based data exfiltration.

Acknowledgements This work has been supported by a predoctoral grant from the Spanish National Cybersecurity Institute (INCIBE) within the program "Ayudas para la Excelencia de los Equipos de Investigación Avanzada en Ciberseguridad" ("Grants for the Excellence of Advanced Cybersecurity Research Teams") with code INCIBEI-2015-27353, the European Commission Horizon 2020 Programme under grant agreement number H2020-SU-DS-2019/883335—PALANTIR (Practical Autonomous Cyberhealth for resilient SMEs & Microenterprises), and the European Commission (FEDER/ERDF).

Compliance with ethical standards

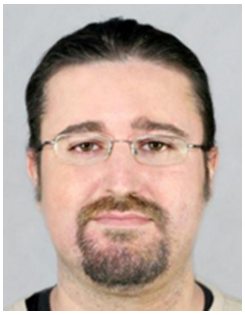
Conflict of interest The authors declare that they have no conflict of interest.

References

1. Afolabi, I., Taleb, T., Samdanis, K., Ksentini, A., Flinck, H.: Network slicing and softwarization: a survey on principles, enabling technologies, and solutions. IEEE Commun. Surv. Tutor. **20**(3), 2429–2453 (2018). https://doi.org/10.1109/COMST.2018.2815638
2. Ahmad, I., Shahabuddin, S., Kumar, T., Okwuibe, J., Gurtov, A., Ylianttila, M.: Security for 5G and beyond. IEEE Commun. Surv. Tutor. **21**(4), 3682–3722 (2019). https://doi.org/10.1109/COMST.2019.2916180
3. Almashhadani, A.O., Kaiiali, M., Carlin, D., Sezer, S.: MaldomDetector: a system for detecting algorithmically generated domain names with machine learning. Comput. Secur. **93**, 101787 (2020). https://doi.org/10.1016/j.cose.2020.101787
4. 5G Americas: 5G at the edge. Tech. rep., 5G Americas (2019). Online: https://www.5gamericas.org/wp-content/uploads/2019/10/5G-Americas-EDGE-White-Paper-FINAL.pdf. Accessed 24 Nov 2020
5. Andrews, J.G., Buzzi, S., Choi, W., Hanly, S.V., Lozano, A., Soong, A.C.K., Zhang, J.C.: What will 5G be? IEEE J. Sel. Areas Commun. **32**(6), 1065–1082 (2014). https://doi.org/10.1109/JSAC.2014.2328098
6. 5G PPP Architecture Working Group: View on 5G architecture. Tech. rep., 5G Infrastructure Public Private Partnership (2020). https://doi.org/10.5281/zenodo.3265031
7. Barakabitze, A.A., Ahmad, A., Mijumbi, R., Hines, A.: 5G network slicing using SDN and NFV: a survey of taxonomy, architectures and future challenges. Comput. Netw. **167**, 106984 (2020). https://doi.org/10.1016/j.comnet.2019.106984
8. Blaise, A., Bouet, M., Conan, V., Secci, S.: Detection of zero-day attacks: an unsupervised port-based approach. Comput. Netw. **180**(January), 107391 (2020). https://doi.org/10.1016/j.comnet.2020.107391
9. Chadwick, D.W., Fan, W., Costantino, G., de Lemos, R., Di Cerbo, F., Herwono, I., Manea, M., Mori, P., Sajjad, A., Wang, X.S.: A cloud-edge based data security architecture for sharing and analysing cyber threat information. Future Gener. Comput. Syst. **102**, 710–722 (2020). https://doi.org/10.1016/j.future.2019.06.026
10. Chirivella-Perez, E., Marco-Alaez, R., Hita, A., Serrano, A., Alcaraz Calero, J.M., Wang, Q., Neves, P.M., Bernini, G., Koutsopoulos, K., Gil Pérez, M., Martínez Pérez, G., Barros, M.J., Gavras, A.: SELFNET 5G mobile edge computing infrastructure: design and prototyping. Software **50**(5), 741–756 (2020). https://doi.org/10.1002/spe.2681
11. Coppolino, L., D'Antonio, S., Mazzeo, G., Romano, L.: A comprehensive survey of hardware-assisted security: from the edge to the cloud. Internet Things **6**, 100055 (2019). https://doi.org/10.1016/j.iot.2019.100055
12. Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S., Zomaya, A.Y.: Edge intelligence: the confluence of edge computing and artificial intelligence. IEEE Internet Things J. (2020). https://doi.org/10.1109/jiot.2020.2984887
13. Dhar, S., Guo, J., Liu, J., Tripathi, S., Kurup, U., Shah, M.: On-device machine learning: an algorithms and learning theory perspective (2019). http://arxiv.org/abs/1911.00623
14. Eskandari, M., Janjua, Z.H., Vecchio, M., Antonelli, F.: Passban IDS: an intelligent anomaly based intrusion detection system for IoT edge devices. IEEE Internet of Things J. (2020). https://doi.org/10.1109/JIOT.2020.2970501
15. Fernández Maimó, L., Perales Gómez, Á.L., García Clemente, F.J., Gil Pérez, M., Martínez Pérez, G.: A self-adaptive deep learning-based system for anomaly detection in 5G networks.

IEEE Access **6**, 7700–7712 (2018). https://doi.org/10.1109/ACCESS.2018.2803446
16. Hard, A., Kiddon, C.M., Ramage, D., Beaufays, F., Eichner, H., Rao, K., Mathews, R., Augenstein, S.: Federated learning for mobile keyboard prediction (2018). http://arxiv.org/abs/1811.03604
17. Huertas Celdrán, A., Gil Pérez, M., García Clemente, F.J., Martínez Pérez, G.: Towards the autonomous provision of self-protection capabilities in 5G networks. J. Ambient Intell. Hum. Comput. **10**(12), 4707–4720 (2019). https://doi.org/10.1007/s12652-018-0848-6
18. Isakov, M., Gadepally, V., Gettings, K.M., Kinsy, M.A.: Survey of attacks and defenses on edge-deployed neural networks. In: 2019 IEEE High Performance Extreme Computing Conference, pp. 1–8 (2019). https://doi.org/10.1109/HPEC.2019.8916519
19. Khan, W.Z., Ahmed, E., Hakak, S., Yaqoob, I., Ahmed, A.: Edge computing: a survey. Future Gener. Comput. Syst. **97**, 219–235 (2019). https://doi.org/10.1016/j.future.2019.02.050
20. Kumar, R., Goyal, R.: On cloud security requirements, threats, vulnerabilities and countermeasures: a survey. Comput. Sci. Rev. **33**, 1–48 (2019). https://doi.org/10.1016/j.cosrev.2019.05.002
21. Laurenza, G., Lazzeretti, R., Mazzotti, L.: Malware triage for early identification of advanced persistent threat activities. Digital Threats **1**(3), 1–17 (2020). https://doi.org/10.1145/3386581
22. Lim, W.Y.B., Luong, N.C., Hoang, D.T., Jiao, Y., Liang, Y., Yang, Q., Niyato, D., Miao, C.: Federated learning in mobile edge networks: a comprehensive survey. IEEE Commun. Surv. Tutor. (2020). https://doi.org/10.1109/COMST.2020.2986024
23. Liu, D., Yan, Z., Ding, W., Atiquzzaman, M.: A survey on secure data analytics in edge computing. IEEE Internet Things J. **6**(3), 4946–4967 (2019). https://doi.org/10.1109/JIOT.2019.2897619
24. Papamartzivanos, D., Gomez Marmol, F., Kambourakis, G.: Introducing deep learning self-adaptive misuse network intrusion detection systems. IEEE Access **7**, 13546–13560 (2019). https://doi.org/10.1109/ACCESS.2019.2893871
25. Pham, Q.V., Fang, F., Ha, V.N., Piran, M.J., Le, M., Le, L.B., Hwang, W.J., Ding, Z.: A survey of multi-access edge computing in 5G and beyond: fundamentals, technology integration, and state-of-the-art. IEEE Access **8**, 116974–117017 (2020). https://doi.org/10.1109/ACCESS.2020.3001277
26. Ren, J., Zhang, D., He, S., Zhang, Y., Li, T.: A survey on end-edge-cloud orchestrated network computing paradigms: transparent computing, mobile edge computing, fog computing, and cloudlet. ACM Comput. Surv. **52**(6), 125 (2019). https://doi.org/10.1145/3362031
27. Sharma, R., Chan, C.A., Leckie, C.: Evaluation of centralised vs distributed collaborative intrusion detection systems in multi-access edge computing. In: 2020 IFIP Networking Conference (Networking), pp. 343–351 (2020)
28. Shi, W.C., Sun, H.M.: DeepBot: a time-based botnet detection with deep learning. Soft Comput. (2020). https://doi.org/10.1007/s00500-020-04963-z
29. Sun, P.J.: Privacy protection and data security in cloud computing: a survey, challenges, and solutions. IEEE Access **7**, 147420–147452 (2019). https://doi.org/10.1109/ACCESS.2019.2946185
30. Tabassi, E., Burns, K.J., Hadjimichael, M., Molina-Markham, A.D., Sexton, J.T.: A taxonomy and terminology of adversarial machine learning—DRAFT. Tech. rep., NIST (2019). https://doi.org/10.6028/NIST.IR.8269-draft
31. Tian, J., Gou, G., Liu, C., Chen, Y., Xiong, G., Li, Z.: DLchain: A covert channel over blockchain based on dynamic labels. In: Information and Communications Security, pp. 814–830 (2020). https://doi.org/10.1007/978-3-030-41579-2_47
32. Wang, X., Han, Y., Wang, C., Zhao, Q., Chen, X., Chen, M.: In-edge AI: intelligentizing mobile edge computing, caching and communication by federated learning. IEEE Netw. **33**(5), 156–165 (2019). https://doi.org/10.1109/MNET.2019.1800286
33. Wang, X., Han, Y., Leung, V.C., Niyato, D., Yan, X., Chen, X.: Convergence of edge computing and deep learning: a comprehensive survey. IEEE Commun. Surv. Tutor. (2020). https://doi.org/10.1109/comst.2020.2970550
34. Xu, X., Liu, X., Yin, X., Wang, S., Qi, Q., Qi, L.: Privacy-aware offloading for training tasks of generative adversarial network in edge computing. Inf. Sci. **532**, 1–15 (2020). https://doi.org/10.1016/j.ins.2020.04.026
35. Yao, H., Gao, P., Zhang, P., Wang, J., Jiang, C., Lu, L.: Hybrid intrusion detection system for edge-based IIoT relying on machine-learning-aided detection. IEEE Netw. **33**(5), 75–81 (2019). https://doi.org/10.1109/MNET.001.1800479
36. Yun, X., Huang, J., Wang, Y., Zang, T., Zhou, Y., Zhang, Y.: Khaos: an adversarial neural network DGA with high anti-detection ability. IEEE Trans. Inf. Forensics Secur. **15**, 2225–2240 (2020). https://doi.org/10.1109/TIFS.2019.2960647
37. Zago, M., Gil Pérez, M., Martínez Pérez, G.: Scalable detection of botnets based on DGA: efficient feature discovery process in machine learning techniques. Soft Comput. **24**, 5517–5537 (2020). https://doi.org/10.1007/s00500-018-03703-8
38. Zago, M., Gil Pérez, M., Martínez Pérez, G.: UMUDGA: a dataset for profiling algorithmically generated domain names in botnet detection. Data Brief **30**, 105400 (2020). https://doi.org/10.1016/j.dib.2020.105400
39. Zago, M., Gil Pérez, M., Martínez Pérez, G.: UMUDGA: a dataset for profiling DGA-based botnet. Comput. Secur. **92**, 101719 (2020). https://doi.org/10.1016/j.cose.2020.101719
40. Zargar, S.T., Takabi, H., Iyer, J.: Security-as-a-Service (SECaaS) in the cloud. Security, Privacy, and Digital Forensics in the Cloud, Chap. 9, pp. 189–200. Wiley, New York (2019). https://doi.org/10.1002/9781119053385
41. Zhang, Y., Huang, H., Yang, L.X., Xiang, Y., Li, M.: Serious challenges and potential solutions for the industrial internet of things with edge intelligence. IEEE Netw. **33**(5), 41–45 (2019). https://doi.org/10.1109/MNET.001.1800478
42. Zhang, C., Ueng, Y., Studer, C., Burg, A.: Artificial intelligence for 5G and beyond 5G: implementations, algorithms, and optimizations. IEEE J. Emerg. Sel. Topics Circ. Sys. **10**(2), 149–163 (2020). https://doi.org/10.1109/JETCAS.2020.3000103
43. Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., Zhang, J.: Edge intelligence: paving the last mile of artificial intelligence with edge computing. Proc. IEEE **107**(8), 1738–1762 (2019). https://doi.org/10.1109/JPROC.2019.2918951

Mattia Zago is a Ph.D. student at the University of Murcia, Spain. He holds a master degree in Computer Science and Engineering from the University of Verona, Italy. His research focuses on Information & Communication Systems Security and Artificial Intelligence, with particular dedication to Machine learning applications to Network Intrusion detection and Response Systems and Behavioral Modeling. More info at https://webs.um.es/mattia.zago.



Manuel Gil Pérez is Associate Professor in the Department of Information and Communication Engineering of the University of Murcia, Murcia, Spain. His scientific activity is mainly devoted to cyber security, including intrusion detection systems, trust management, privacy-preserving data sharing, and security operations in highly dynamic scenarios. He received M.Sc. and Ph.D. degrees (latter with distinction) in Computer Science from the University of Murcia. He is co-author of 70+ scientific publications in journals and conference papers, as well as an active member on different national and international research projects. More info at https://webs.um.es/mgilperez.

Gregorio Martínez Pérez is Full Professor in the Department of Information and Communications Engineering of the University of Murcia, Spain. His scientific activity is mainly devoted to cybersecurity, privacy and networking. He is working on different national and European IST research projects (25 in the last decade) on these topics, being Principal Investigator in most of them. He has published 160+ papers in national and international conference proceedings, magazines and journals. More info at https://webs.um.es/gregorio.