big data and cognitive computing

MDPI

Article

The Next Generation Cognitive Security Operations Center: Adaptive Analytic Lambda Architecture for Efficient Defense against Adversarial Attacks

Konstantinos Demertzis [1,*], Nikos Tziritas [2], Panayiotis Kikiras [3], Salvador Llopis Sanchez [4] and Lazaros Iliadis [1]

[1] Department of Civil Engineering, School of Engineering, Democritus University of Thrace, 67100 Xanthi, Greece; liliadis@civil.duth.gr

[2] Research Center for Cloud Computing, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518000, China; nikolaos@siat.ac.cn

[3] Department of Computer Science, School of Science, University of Thessaly, 35131 Lamia, Greece; kikirasp@uth.gr

[4] Communications Department, Universitat Politecnica de Valencia, 46022 Valencia, Spain; salllosa@masters.upv.es

* Correspondence: kdemertz@fmenr.duth.gr or kdemertzis@uth.gr Tel.: +30-694-824-1881

Received: 25 November 2018; Accepted: 7 January 2019; Published: 10 January 2019

Abstract: A Security Operations Center (SOC) is a central technical level unit responsible for monitoring, analyzing, assessing, and defending an organization's security posture on an ongoing basis. The SOC staff works closely with incident response teams, security analysts, network engineers and organization managers using sophisticated data processing technologies such as security analytics, threat intelligence, and asset criticality to ensure security issues are detected, analyzed and finally addressed quickly. Those techniques are part of a reactive security strategy because they rely on the human factor, experience and the judgment of security experts, using supplementary technology to evaluate the risk impact and minimize the attack surface. This study suggests an active security strategy that adopts a vigorous method including ingenuity, data analysis, processing and decision-making support to face various cyber hazards. Specifically, the paper introduces a novel intelligence driven cognitive computing SOC that is based exclusively on progressive fully automatic procedures. The proposed λ-Architecture Network Flow Forensics Framework (λ-NF3) is an efficient cybersecurity defense framework against adversarial attacks. It implements the Lambda machine learning architecture that can analyze a mixture of batch and streaming data, using two accurate novel computational intelligence algorithms. Specifically, it uses an Extreme Learning Machine neural network with Gaussian Radial Basis Function kernel (ELM/GRBFk) for the batch data analysis and a Self-Adjusting Memory k-Nearest Neighbors classifier (SAM/k-NN) to examine patterns from real-time streams. It is a forensics tool for big data that can enhance the automate defense strategies of SOCs to effectively respond to the threats their environments face.

Keywords: network flow forensics; adversarial attacks; malware traffic analysis; security operations center; cognitive cybersecurity intelligence; lambda architecture

1. Introduction

With an ever-increasing cybersecurity threat landscape to interconnected or networked devices, and since the volume of data is growing exponentially, it is more important than ever for critical

infrastructures and organizations to be strengthened with intelligent driven security managing and monitoring tools. Using the right combination of these intelligent centralized tools and big data technologies allows classifying risks with high accuracy across network infrastructures to identify sophisticated attacks. Nevertheless, current SOCs focus mostly on human experience and the opinion of experts to evaluate and minimize potential cyber threats. On the other hand, traditional signature-based security systems are unable in most cases to identify evolving threats such as zero-day malware or they produce a vast number of false alarms, thus are proven ineffective as security management tools. The implementation and usage of alternative, more innovative and more effective intelligent methods with fully automated aptitudes appear is necessary to produce an-up-to date SOC that can handle security incidents.

Accordingly, SOCs are being forced to consider new ways to boost their cyber defenses such as cloud strategies, big data analytics and artificial intelligence technologies that are emerging as the frontrunner in the fight against cyber-crime. With fully self-governed systems that mimic the functioning of the human brain and help to improve decision-making with minimum human interference, a Next Generation Cognitive Computing SOC (NGC2SOC) is in a far better place to strengthen and reinforce cybersecurity strategies. The ultimate purpose of NGC2SOC comprises sophisticated intelligence driven tactics for real-time investigation of both known and unknown vulnerabilities, immediate access, evidence visualization and additional advanced tools or practices that reduce the potential risk in critical assets combined with a completely automated reinstatement of cybersecurity problems.

Machine learning is a practice used to develop sophisticated representations and systems that produce dependable, repeatable decisions and discover unseen or hidden patterns through learning from historical data. In these models, the training and test data are expected to be produced from identical although probably unidentified distributions, thus they have been very sensitive to slight changes in the input or a series of specific transformations [1]. Most of those sensitivities under certain circumstances may lead to altering the behavior of the machine learning algorithms. Specifically, security of machine learning systems is vulnerable to crafted adversarial examples, which may be imperceptible to the human eye, but can lead the model to misclassify the output. In recent times, different types of adversaries based on their threat model leverage these vulnerabilities to compromise a machine learning system where adversaries have high incentives.

An adversarial attack is an attempt to maliciously operate the input data or manipulate specific weaknesses of machine learning procedures to compromise the entire security system. For example, a classification process by a trained neural network classifier decides which class a new remark fits based on a training set of data covering remarks whose class association is known. The classification threshold is imperfect and an appropriately designed and implemented adversarial attack, which corresponds to a modified input that may come from a modified dataset, can lead the algorithm to a wrong solution (wrong class). This is because the neural networks operate on high-dimensional data, they are sensitive to overfitting, they can be too linear and they are characterized by the inherent uncertainty of their predictions.

To understand the security properties of learning algorithms in adversarial settings, one should address the following main issues:

- identifying potential vulnerabilities of machine learning algorithms during learning and classification;
- devising appropriate attacks that correspond to the identified threats and evaluating their impact on the targeted system; and
- proposing countermeasures to improve the security of machine learning algorithms against the considered attacks.

In general, there are two defense strategies against adversarial attacks. First is the reactive strategy, which consists of training another classifier, which should be constructed based on the variety of the execution mode and on the restrictions' settings that can lead to dissimilar decision boundaries, even if all other constraints remain steady. For example, different classifiers should be chosen with multiple levels of diversity that use dissimilar functional settings and diverse training

sets, thus permitting dissimilar decision boundaries to be formed, and that can be combined to reduce the overall error. The second is the proactive strategy, which relies on implementing suitable precautionary training, capable of establishing the exact decision boundaries. An investigation that considers the training process of the learning model should try to discover the optimum weights. The weight vector is a very important parameter, as it is used in the development of defining the confidentiality of classifiers, and the confidence of the pattern recognition process. For example, in the situation of higher weights, it is an important request of the classification process of how they regulate the class boundaries of the general prototype. Hence, it is extremely important to provide robustness to machine learning algorithms against these adversaries.

This paper proposes the development of an innovative λ-Architecture Network Flow Forensics Framework (λ-NF3) to network traffic analysis, demystification of malware traffic and encrypted traffic identification for efficient defense against adversarial attacks. The λ-NF3 is an effective and accurate network administration system that offers intelligent network flow forensics methods, aiming to be used by NGC2SOCs that can work without the need of human experience and the opinion of experts to evaluate and minimize potential cyber threats. A basic innovation of the proposed methodology is the combination of two sophisticated algorithms for the first time in a hybrid machine learning framework. The proposed framework employs a specific version of the Lambda architecture combined with Extreme Learning Machine with Gaussian Radial Basis Function kernel (ELM/GRBFK) for the batch data classification and k-NN Classifier with Self Adjusting Memory (SAM/k-NN) to investigate real time data streams. Lambda architecture was chosen, as, in multifactorial problems of high complexity of large datasets such as the one under consideration, the outcomes of the estimation are multi-variable, especially with respect to analysis and integration of network data flows. This implementation follows a reactive cyber security strategy for dealing with adversarial attacks as it combines training two diametrically opposite classifiers to detect incoming potential threats and to discard them. In addition, it is important to highlighted that the proposed novel scheme offers high learning speed, ease of execution, minimal human involvement and minimum computational power and resources.

The remainder of this paper is organized as follows. Section 2 presents the literature review on machine learning approaches have used in the traffic analysis, how Lambda architecture improves the overall accuracy of a big data model and some interesting methods to hardening a machine learning system against adversarial attacks. Section 3 defines the proposed framework. Section 4 outlines the methodology. Section 5 present the datasets. Section 6 explains the results. Section 7 present the conclusions.

2. Literature Review

Adversarial attacks have been recorded against spam filtering, where spam messages are obscured through spelling mistakes [2]; in computer networks, where malware code masquerade as benign network packets [3]; in antivirus software, where malicious program pass the signature detection test [4]; and in biometric recognition, where false biometric features may be alternated to mimic an authentic user (biometric spoofing) [5]. The network flow classification and categorization problem require a vast amount of computing resources [6]. In addition, the exponential increase of collected daily network data has led to the need for big data storage applications that should be designed for high capacity, low latency, and rapid analytics. In addition, the velocity of data and the necessity of real-time analysis, combined with the variety of both structured and non-structured data forms, present many challenges including scalability and storage bottleneck, noise gathering, false correlation, supplementary endogeneity, and measurement errors. However, the biggest challenge in the analysis of network flow data, which is a big data processing problem, is the performance of pattern recognition and knowledge mining by employing intelligent systems with proper architectures [7].

In addition, SOC staff are assisted by visual tools when studying big data. Their clarification of the genuineness on the screen may vary due to their familiarity and skills. An essential request of the

efficiency is to maximize operators' cyber condition alertness by adopting expressive visualization tools as part of an all-inclusive decision-support method [8].

The λ-NF3 is an effective and innovative intelligence-driven cyber security method. This study has emerged after extensive and long-term research about the network forensics process with cyber-security methodologies and specifically about the network traffic analysis, demystification of malware traffic and encrypted traffic identification [9–17]. Significant work has been done using various machine learning methods in various domains.

For example, one study demonstrates that vulnerabilities can be predicted using an SVM model based on a set of code metrics for a specific Android application [18]. The classification model exhibits good performance in terms of both accuracy and precision. However, this study applies to a limited pool of applications and few Android versions. In addition, Shabtai et al. [19] proposed a heuristic approach to static analysis of Android applications based on matching suspicious applications with the predefined malware models. Static models are built from Android capabilities and Android Framework API call chains used by the application. All the analysis steps and model construction are fully automated. However, the proposed method has smaller detection coverage with randomly chosen malware models.

In addition, in [20], the authors proposed an inter-application communication tool that detects application communication vulnerabilities. The proposed model can be used by developers to analyze their own applications before release, by application reviewers to analyze applications in the Android market, and by end users. The authors analyzed 20 applications and found 34 exploitable vulnerabilities; 12 of the 20 applications have at least one vulnerability. This shows that applications can be vulnerable to attack and that developers should take precautions to protect themselves from these attacks. Burguera et al. [21] proposed a behavior-based malware detection system, while Glodek at al. [22] a permissions-based malware detection system; however, the classification performances of these systems are severely affected by limited supervised information and unknown applications.

On the other hand, Zhang et al. [23] developed a new method to tackle the problem of unknown applications in the crucial situation of a small supervised training set. The proposed method possesses the superior capability of detecting unknown flows generated by unknown applications and utilizing the correlation information among real-world network traffic to boost the classification performance. A theoretical analysis is provided to confirm the performance benefit of the proposed method. Moreover, the comprehensive performance evaluation conducted on two real-world network traffic datasets shows that the proposed scheme outperforms the existing methods in the critical network environment.

Malware attacks are increasingly popular attack vectors in online crime. As trends and anecdotal evidence show, preventing these attacks, regardless of their opportunistic or targeted nature, has proven difficult: intrusions happen, and devices get compromised, even at security-conscious organizations. Therefore, an alternative line of work has focused on detecting and disrupting the individual steps that follow an initial compromise and that are essential for the successful progression of the attack. Several approaches and techniques have been proposed to identify the Command and Control (C2) channel that a compromised system establishes to communicate with its controller. The success of C2 detection approaches depends on collecting relevant network traffic. As traffic volumes increase, this is proving increasingly difficult.

For example, Gardiner et al. [24] analyzed current approaches of ISP-scale network measurement from the perspective of C2 detection, discussed several weaknesses that affect current techniques and provided suggestions for their improvement. Hsu et al. [25] proposed an innovative structure for detecting botnets in real time based on performance metrics to investigate whether a suspicious server is a fast-flux bot. The most innovative part of this approach is the fact that it works in either passive or active mode. Valuations show that the proposed solution is a promising method that can identify the botnet's activities without noteworthy performance deprivation, however the method fails in the situation of encrypted communication of the compromised machines of the botnet.

In addition, Haffner et. al. [26] employed an automated method to export the payload content from network flow of real-time applications and used several machine learning models to categorize the network traffic. The proposed method is time consuming and requires high CPU utilization. Furthermore, Holz et al. [27] developed a heuristic approach to calculating some properties that identify some fast-flux botnets. This is a passive method to locate obsolete botnets and fails to investigate dynamic fast-flux botnets based on sophisticated techniques. Almubayed et al. [28] presented a very interesting method to extract several features from the encrypted traffic of the Tor network. These features are appropriate and can classify the Tor traffic with very high accuracy.

On the other hand, several optimal and novel applications have been done in applying Lambda architecture [29,30]. For example, Kiran et al. [31] presented a cost-optimized lambda architecture that combines online and batch data processing to handling a huge volume of sensor data. Both procedures can produce effective data accumulations, combinations or aggregations that are easier to analyze for identifying hidden patterns. It is also a promising method that reduces significantly the processing time and the resource requirements. Moreover, Yamato et al. [32] used a lambda architecture to analyze data from IoT sensors. It is a data analytics framework that uses incremental learning techniques to identify anomalies in real time.

The automatically updating learning model improves analysis accuracy and is a promising method to defend against adversarial attacks. A new valuation procedure that is resilient to face these attacks was proposed by Yong et al. [33]. Specifically, the authors, to respond to injection attacks and adversarial additive and multiplicative errors, proposed a method to split the dataset into uncertainness subsets that lead to a manageable optimization result. In addition, Chong et al. [34] proposed a combination method of two algorithms to defend against adversarial attacks. In the first stage, an effective algorithm uses a finite window of measurements to reconstruct the initial state. In the second stage, a different algorithm intervenes to the exact state appreciation . Finally, Chen et al. [35] introduced a security regularization term that contemplates the circumvention cost of feature handlings by attackers to increase the system security.

3. Description of the Proposed Framework

3.1. Network Forensics

Network forensics is a progressive procedure involving the monitoring and analysis of network traffic for information congregation, legal investigation, or intrusion detection and prevention. In addition, network forensics arrange provisional and momentary evidence in an unpredictable and dynamic environment such that network traffic is transmitted and then lost. An imperative subsection of network forensics is the traffic classification process, which is an automated procedure to classify network patterns according to numerous constraints into several traffic classes. The main method to recognize and classify network traffic with high accuracy and precision is the classification process based on the payload [36].

Serious weaknesses of these methods are the complexity and their requirements in terms of computational resources. In addition, cyber security expert opinion is required to differentiate provided services and implement appropriate security policies. Even the most sophisticated forecast method that relies on Deep Packet Inspection (DPI) is time-consuming, does not produce accurate results and suffers high false alarm rates [37].

To summarize, network flow forensics methods depend on the availability of various system resources, need supervision from a network engineer with advanced skills in cybersecurity and nevertheless fail totally to identify zero-day exploitations.

Malicious botnets have become the most dangerous threat of the Internet today using advanced techniques to obfuscate the network communication aspects involved in their phishing schemes, malware delivery or other criminal enterprises. Thus, malware traffic analysis is the primary method of botnet investigation and identification of the command and control (C2) infrastructures associated with these activities. The most sophisticated types of malware are seeking network communication aspects in botnet establishment and operation with the C2 isolated servers via an obfuscated

communication layer, based on the fuzzy construction of the Tor network. The Tor network produces traffic similar to the normal encryption traffic of the HTTPS protocol, making the identification procedure extremely difficult [38].

Since cyber systems' security is a multifaceted procedure, SOC management cannot be based only on the passive-mode signature-based defense applications that are ineffective on zero-day attacks. The discovery and identification of a penetration or interruption in the network should be a self-acting and nearly real-time procedure, which would offer an imperative advantage to the administrators. In this point of view, the use of more effective methods of network supervision, with capabilities of automated control in network traffic analysis, demystification of malware traffic and encrypted traffic identification is important to estimate the behavior of malware, the purpose of attacks and the damage caused by malware activity.

3.2. Batch Processing

Batch data are usually datasets collected during some transactions or processes for a certain time period and characterize the systems functionality. Processing these data using conventional data mining or machine learning methods assumes that they are available and can be accessed simultaneously without any limitation in terms of their processing or analysis time. It should be noted that these data are susceptible to noise, their classification process has a significant cost, and they require serious hardware infrastructure for safe storage and general handling.

Batch processing is the implementation of a sequence of procedures from batch data. This processing can be concluded or scheduled in the time period when the computing resources are less busy. Similarly, it evades wasting system resources with manual intercession and management, keeping high overall rate of utilization. It permits the system to use diverse processes for collaborating works and separation tasks, thus reducing the storage overhead and shifting bottlenecks. However, the batch processing also has numerous drawbacks, for example, users are unable to terminate a progression and must wait until the execution completes.

With the new technologies that have dominated our lives over the last few years and especially with the constant rise of Internet sensors and actuators, the volume of data generated by devices is constantly increasing. The rising field of real-world implementations produces data streams at a cumulative percentage, needing large-scale and real-time processing.

3.3. Stream Processing

Data streams are endless data that are produced by multiple network infrastructures, such as sensors, IoT equipment, etc. A typical example for streaming data is the data that can be collected from network monitoring queries at high traffic rates. Streaming data should be handled in sequence and incrementally or over sliding time windows. Due to their reliance on strict time constraints and their more general availability, they are selected for detailed and specialized data processing techniques that can lead to multiple levels of revelation of the hidden knowledge they may contain. In addition, these data need to be processed without accessing all the rest of the data. In addition, it should be considered that concept drift may occur in the data, which means that the stream properties may transform and alternate over time. It is frequently used in big data projects, in which data streams are quickly produced by several dissimilar sources.

Streaming data and data generated by dynamic environments have lead to some of the most robust research areas of the new era. stream processing techniques are used by machine learning applications on data streams for real time analysis and knowledge extraction under displacement and feedback environments.

In the case of stream processing techniques by machine learning, the algorithms are controlled by a variety of possible shifting modes and constraints related to memory consumption, resource limitation and processing time. In this category, the available data are scaled in a sequential order and used for training and forecasting by calculating the error in each iteration. The aim of the algorithms in this category is to minimize the cumulative error for all iterations. We consider that the

intention of supervised learning using the square loss function is to minimize the empirical error calculated by the following function [39]:

$$I_n[w] = \sum_{j=1}^{n} V\left(\langle w, x_j\rangle, y_j\right) = \sum_{j=1}^{n} \left(x_j^T w - y_j\right)^2 \quad (1)$$

where $x_j \in R^d, w \in R^d$ and $y_j \in R$. Let there be a data table of $Xi \times d$ and a target values table of dimensions $\Upsilon i \times 1$ as they are defined after the entrance of the first i data points.

Let us suppose that the covariance table $\Sigma_i = X^T X$ is reversible, and $f^*(x) = \langle w^*, x\rangle$ is the ideal result for the linear least squares problem, as shown in Equation (2):

$$w^* = (X^T X)^{-1} X^T \Upsilon = \Sigma_i^{-1} \sum_{j=1}^{i} x_j y_j \quad (2)$$

The calculation of the table $\Sigma_i = \sum_{j=1}^{i} x_j x_j^T$ has a time complexity of $O(id^2)$. Reversing the $d \times d$ table has a time complexity of $O(d^3)$, whereas the rest of the multiplication requires time complexity of $O(d^2)$, producing an overall complexity of $O(id^2 + d^3)$. If we consider that n is the set of points in the dataset $i = 1,2, \ldots, n$ and it is essential to recalculate the result after the arrival of each new data vector, we obtain a total complexity $O(n^2 d^2 + nd^3)$ [39,40].

It is important here to mention that a machine learning stream processing is appropriate in cases where it is required to dynamically adapt the procedure to new standards or data, or when the streams are produced as a function of time, as in the case of the research of the adversarial attacks.

3.4. The Proposed Approach

The need to extract information from extensive networking flows in real time is a big data challenge, such as those that re-established the prototyping architectures of big data. Big data architectures include mechanisms for ingesting, protecting, processing, and transforming data into big data structures. In addition, these architectures typically comprise an examination of data lakes (batch processing), real-time analyses (streaming processing), predictive analytics from unstructured data and machine learning tools that analyze data with low latency [31].

The analysis of very large volumes of data is time consuming and cannot be completed in real time. Abundant data storage that works with the entire dataset to store the results of the queries for future use is frequently required. One serious disadvantage to this method is that it introduces latency.

The lambda architecture faces this problem by producing two pathways for data flow. A batch layer (cold path) includes all inbound data in their raw format and achieves batch processing on the data. The outcome of this analysis is stored and deposited as a batch view. In addition, a speed layer (hot path) analyzes unbounded streams of data in real time. The hot path is planned for low latency, at the expense of accuracy [32]. Figure 1 is a depiction of the lambda architecture.

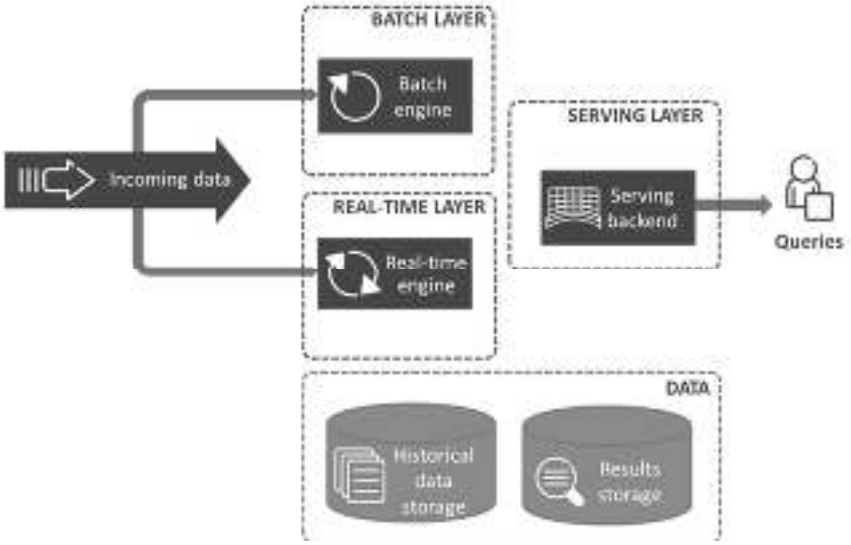

Figure 1. Lambda architecture (www.ericsson.com).

The lambda architecture has been designed to balance latency, throughput, and fault tolerance using the cold path to provide complete and accurate views of historical data. At the same time, it uses the hot path to provide real time data stream analysis of new inputs. Finally, an additional element that enhances the process and adds accuracy in the entire model is that the two projection outputs can be joined before the final data presentation or the final decision.

The algorithmic approach of the λ-NF3 in the first phase includes the feature extraction procedure from network flow. In the second phase, these features are analyzed by both classifiers to minimize the possibility of being deceived by adversarial attacks. Both results are merged with a bias to the cold path (batch processing) because it allows good audit trail, although the real-time processing is more difficult for auditing.

A depiction of the λ-NF3 process is shown in Figure 2.

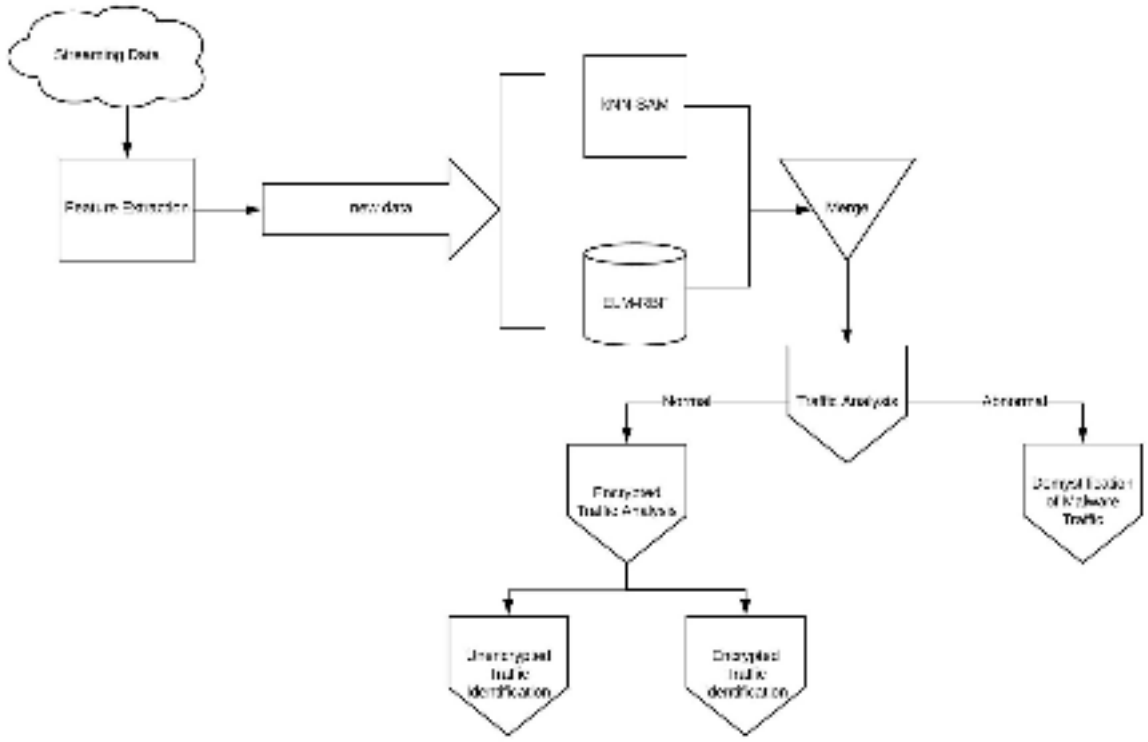

Figure 2. The algorithmic process of the proposed λ-NF3.

The first investigation is whether the traffic is normal or abnormal (network traffic analysis). If the traffic is abnormal, it will be further analyzed (malware traffic demystification) for the purpose of identifying the specific abnormality (botnet, crimeware, Advanced Persistent Threat (APT) attack, CoinMiner, etc.). Besides, if the traffic is normal, it will be further analyzed to inspect whether the application or protocol uses non-encrypted traffic (File Transfer Protocol (FTP), Hypertext Transfer Protocol (HTTP), Domain Name System (DNS), Simple Mail Transfer Protocol (SMTP), etc.) or encrypted traffic (encrypted traffic identification), as well as which protocol that it uses (The onion router (Tor), Secure Shell (SSH), Secure Sockets Layer Web (SSLweb), Secure Sockets Layer Peer-to-Peer (SSLP2P), Secure Copy Protocol (SCP), Skype, etc.).

A depiction of the classification process of the proposed λ-NF3 is presented in the following Figure 3.

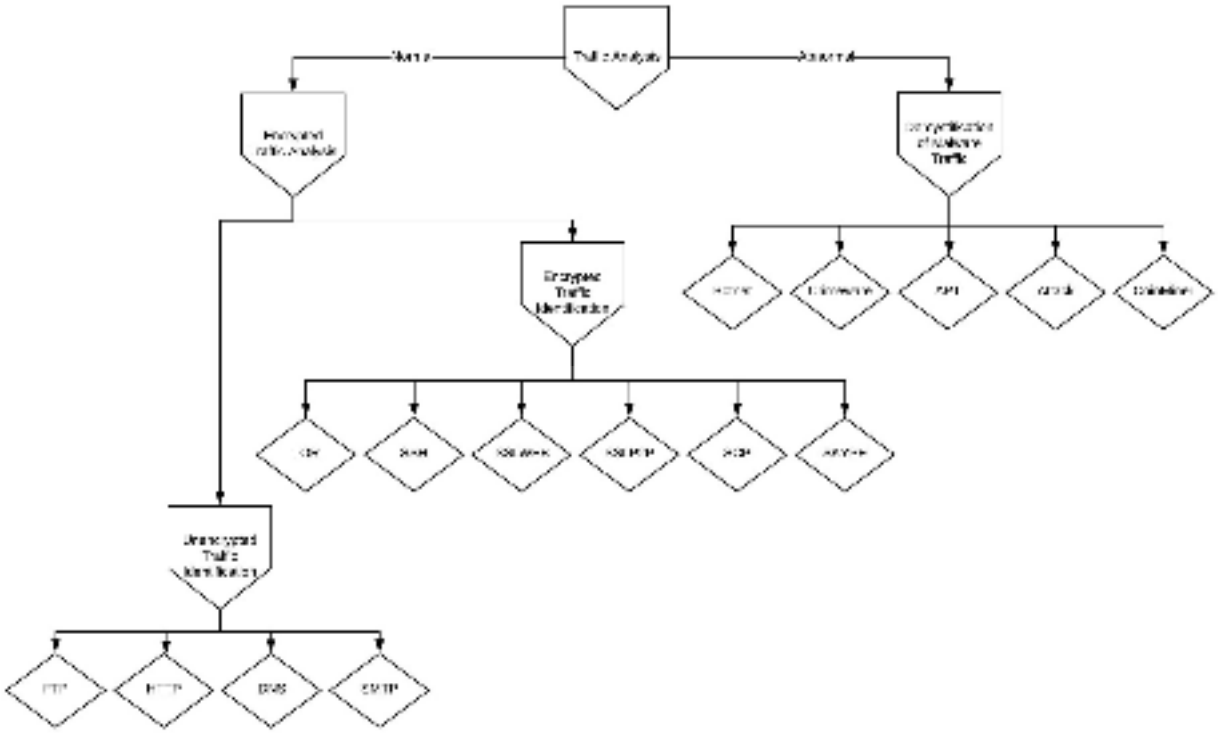

Figure 3. The classification process of the proposed λ-NF3.

The overall process is presented in Algorithm 1.

Algorithm 1 The λ-NF3 Algorithm

Inputs: Input new network traffic data $\mathbf{D_1}$

Step 1: % *Features Extraction*

Feature extraction from network flow

Step 2: % *Make a prediction*

Use the pretrained ELM-RBF classifier using $\mathbf{D_1}$ to produce the prediction $\mathbf{M_1}$

Use the streaming SAM/k-NN classifier using $\mathbf{D_1}$ to produce the prediction $\mathbf{M_2}$

Assign weights of 0.60 to $\mathbf{M_1}$

Merge the two predictions $\mathbf{M_1}$ and $\mathbf{M_2}$ into M

Output from traffic analysis:

if Abnormal

Malware Traffic Demystification

for a class label **Botnet**, **Crimeware**, **APT**, **Attack**, **CoinMiner**

else

if Encrypted

Encrypted Traffic Identification

for a class label **Tor**, **SSH**, **SSLweb**, **SSLP2P**, **SCP**, **Skype**

else

for a class label **FTP**, **HTTP**, **DNS**, **SMTP**

end if

end if

Outputs: Class label for each new data $\mathbf{D_1}$

4. Methodology

The λ-NF3 was developed by employing a general concept of coupling different types of sophisticated algorithms with significant diversity in their operation and configuration mode, with

different architectures, requiring different realizations, hyper-parameter settings and training techniques. These algorithms are presented below.

4.1. Extreme Learning Machines for Batch Data

An ELM is a type of Single-Hidden Layer Feed Forward Neural Network (SLFFNN) [41] with N hidden neurons. The most impressive characteristic of the ELMs is the fact that the input weights and the bias in the hidden layer are assigned randomly [42]. In addition, an ELM can precisely learn K samples, thousands of times greater [43] than a back-propagation feed forward neural network because parameters such as stopping criterion, learning rate and learning epochs do not need to be tuned.

The mathematical background of the ELM is presented in [41–43]. Generally, the input data in an ELM is related to a random L future space with a training set N, where $(x_i, t_i), i \in [\![1, N]\!]$ with $x_i \in R^d$ and $t_i \in R^c$. The output is calculated as follows [41–43]:

$$f_L(x) = \sum_{i=1}^{L} \beta_i h_i(x) = h(x)\beta \quad i \in [\![1, N]\!] \tag{3}$$

Vector matrix $\beta = [\beta_1, \ldots, \beta_L]^T$ is the outcome that includes the weight matrix from the hidden and output layers. $h(x) = [g_1(x), \ldots, g_L(x)]$ is the outcome of the hidden layer for the input x, and $g_1(x)$ is the outcome of the ith neuron. If $\{(x_i, t_i)\}_{i=1}^{N}$ is a training set, then $H\beta = T$, where $T = [t_1, \ldots, t_N]^T$ is learning problem with T the outcome and H the hidden layer of an ELM that is calculated as follow:

$$H(\omega_j, b_j, x_i) = \begin{bmatrix} g(\omega_1 x_1 + b_1) & \cdots & g(\omega_l x_1 + b_l) \\ \vdots & \ddots & \vdots \\ g(\omega_1 x_N + b_1) & \cdots & g(\omega_l x_N + b_l) \end{bmatrix}_{N \times l} \tag{4}$$

This research uses ELM/GRBFK. The Gaussian kernel is as follows:

$$K(u, v) = \exp(-\gamma | |u - v| |2) \tag{5}$$

ELM is an important approach for handling and analyzing big data as it requires the minimum training time relative to the corresponding engineering learning algorithms; it does not require fine manipulations to determine its operating parameters; and it can determine appropriate output weights towards the most effective resolution of a problem. Most importantly, they have the potential to generalize, in contrast to corresponding methods that adjust their performance based solely on their training dataset. It is also obvious that the emerging use of ELM in big data analysis creates serious prerequisites for complex systems' development by low cost machines.

4.2. kNN Classifier with Self Adjusting Memory for Streaming Data

The SAM/k-NN is an artificial intelligence algorithm that is biologically inspired from multiple human memory systems, specifically short- and long-term memory [44]. Short-Term Memory (STM) is the capacity for holding, but not manipulating, a small quantity of information for a short time period. It is defined in contrast to Long-Term Memory (LTM) that is the stage of the memory model where informative knowledge is held indefinitely.

Recurrent reactivations are the primary mechanism that encode the memory information culminating in the spreading of information to supplementary locales and integration of new knowledge. Generally speaking, information from STM are conveyed to LTM in the process of the transformation over time of knowledge, which is called memory consolidation. For instance, once someone learns how to ride a bike, it is never forgotten because it is stored within the LTM, making it impossible to be lost. The SAM architecture is partly inspired by this model. For example, the general statement of new inputs (streaming data) is more related for current estimates that can be associated with temporal trends or time-based events. On the other hand, the batch processing from historical data can lead to much better prediction results, while offering generalization.

The SAM architecture is described in Figure 4.

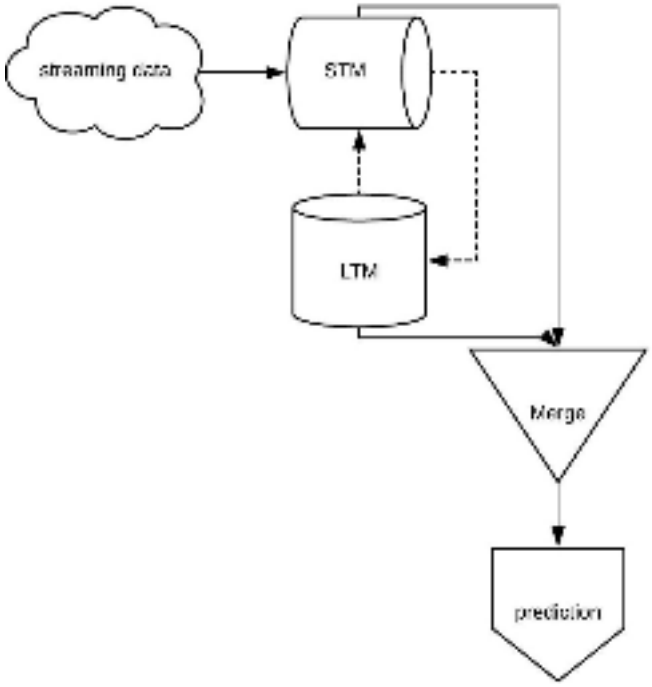

Figure 4. SAM architecture.

Memories are represented by the sets MST, MLT and MC. Each memory is a subset of $R^n \times \{1, \ldots, c\}$ with different length that fluctuates throughout the adjustment procedure. STM signifies the present idea and is an active sliding window that contains the most recent samples m of the data streams [44]:

$$M_{ST} = \{(x_i, y_i) \in R^n \times \{1, \ldots, c\} \,|i = t - m + 1, \ldots, t\} \quad (6)$$

The LTM includes all previous information that is not opposing those of the STM. Different from the STM, the LTM is a set of p points:

$$M_{LT} = \{(x_i, y_i) \in R^n \times \{1, \ldots, c\} \,|i = 1, \ldots, p\} \quad (7)$$

The union of both memories is called Combined Memory (CM) and defined as:

$$\text{MC} = \text{MST} \cup \text{MLT} \quad (8)$$

Every set induces a distance weighted k-NN classifier:

$$R^n \times \{1, \ldots, c\}, kNN_{M_{ST}}, kNN_{M_{LT}}, kNN_{M_C} \quad (9)$$

The function of the kNN algorithm has the aim of assigning a class-label for a given data-point x based on a set $Z = \{(x_i, y_i) \in R^n \times \{1, \ldots, c\} \,|i = 1, \ldots, n\}$:

$$kNN_Z(x) = argmax \left\{ \sum_{x_i \in N_k(x,Z) | y_i = \hat{c}} \frac{1}{d(x_i, x)} | \hat{c} = 1, .., c \right\} \quad (10)$$

where $d(x_i, x)$ is the Euclidean distance between two data points and $N_k(x, Z)$ returns the k nearest neighbors of x in Z. The SAM/k-NN model was introduced by Losing et al. [44].

The implementation of this algorithm as a data stream categorization model is based on the general assumption that new data are more relevant to current forecasts, but prior knowledge is required to properly rank them. The optimal combination of the two processing levels can minimize errors and increase classification precision. The implementation of this model, which provides knowledge transfer potential, is an effective and real time forensics tool to cyber or adversarial attacks identification.

5. Data

The detailed extraction process [45] that includes the appropriate features, which can identify network attacks from the network flow, is analytically described in [46]. It should be noted that this extraction procedure is also enriched by some innovative representation practices for data structures and alteration that introduce the Pandas tool for data manipulation in Python.

The Network Traffic Analysis (NTA) binary dataset that contains of 30 independent variables and 2 classes (normal or abnormal); the Demystification of Malware Traffic (DMT) multiclass dataset that contains 30 independent variables and 5 malware classes (Botnet, Crimeware, APT, Attack and CoinMiner); the Encrypted Traffic Analysis (ETI) binary dataset that includes 30 independent variables and 2 classes (encrypted or non-encrypted); the Encrypted Traffic Identification (EnTI) multiclass dataset that encompasses 30 independent variables and 6 classes that represent encrypted protocols (Tor, SSH, SSLweb, SSLP2P, SCP, and Skype); and the Unencrypted Traffic Identification (UTI) multiclass dataset that comprises 30 independent variables and 4 classes of unencrypted network protocols (FTP, HTTP, DNS, and SMTP), were determined to create extremely complex situations that can potentially include the most likely cases that can be detected in a network infrastructure and that are suitable to train the proposed λ-NF3 [47].

The full list of the 30 data features is detailed in Table 1.

Table 1. Data Features.

ID	Name	Interpretation
1	srcip	The source IP address of the flow.
2	srcport	The source port number of the flow.
3	dstip	The destination IP address of the flow.
4	dstport	The destination port number of the flow.
5	total_fpackets	The total number of packets travelling in the forward direction.
6	total_bpackets	The total number of packets travelling in the backward direction.
7	min_fpktl	The minimum packet length (in bytes) from the forward direction.
8	max_fpktl	The maximum packet length (in bytes) from the forward direction.
9	min_bpktl	The minimum packet length (in bytes) from the backward direction.
10	max_bpktl	The maximum packet length (in bytes) from the backward direction.
11	min_fiat	The minimum interarrival time (in microseconds) between two packets.
12	max_fiat	The maximum interarrival time (in microseconds) between two packets.
13	min_biat	The minimum interarrival time (in microseconds) between two packets.
14	max_biat	The maximum interarrival time (in microseconds) between two packets
15	duration	The time elapsed (in microsec) from the first packet to the last packet.
16	min_active	The minimum duration (in microseconds) of a sub-flow.
17	max_active	The maximum duration (in microseconds) of a sub-flow.
18	min_idle	The minimum time (in microseconds) the flow was idle.
19	max_idle	The maximum time (in microseconds) the flow was idle.
20	sflow_fpackets	The average number of forward travelling packets in the sub-flows.
21	sflow_fbytes	The average number of bytes, travelling in the forward direction.
22	sflow_bpackets	The average number of backward travelling packets in the sub-flows.
23	sflow_bbytes	The average number of bytes, travelling in the backward direction.
24	fpsh_cnt	The number of times the PSH flag was set for packets travelling in the forward direction.
25	bpsh_cnt	The number of times the PSH flag was set for packets travelling in the backward direction.
26	furg_cnt	The number of times the URG flag was set for packets travelling in the forward direction.
27	burg_cnt	The number of times the URG flag was set for packets travelling in the backward direction.

28	total_fhlen	The total header length (network and transport layer) of packets travelling in the forward direction.
29	total_bhlen	The total header length (network and transport layer) of packets travelling in the backward direction.
30	dscp	Differentiated services code point, a field in the IPv4 and IPv6 headers.
31	label	Class

The detailed collection procedure is analytically described in [48].

6. Results

In all simulations, the testing hardware and software conditions are listed as follows: Laptop Intel-i7 2.4 G CPU, 16 G DDR3 RAM, Ubuntu 18.04 LTS, Anaconda Python Data Science Platform and TensorFlow Python environment.

6.1. Batch Data Classification Performance

The classification performance in the batch process was measured by the development of a Confusion Matrix (CM) and then calculating the True Positive Rate (TPR), the True Negative Rate (TNR) and the Total Accuracy (TA), as defined by Equations (11)–(13), respectively [49,50]:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{11}$$

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{12}$$

$$\mathrm{TA} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{N}} \tag{13}$$

In addition, the Precision (PRE), Recall (REC) and F-Score indices allowed the distinctive and irrefutable evaluation of the model. These matrices are defined in Equations (14)–(16), respectively [49,50]:

$$\mathrm{PRE} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{14}$$

$$\mathrm{REC} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{15}$$

$$\mathrm{F-Score} = 2\mathrm{X}\frac{\mathrm{PRE\ X\ REC}}{\mathrm{PRE}\ + \mathrm{REC}} \tag{16}$$

Ten-fold cross validation (10_FCV) was employed to measure performance indices. Tables 2–6 present the outcomes of the λ-NF3 method and the equivalent results from competitive algorithms (Support vector Machine (SVM), Multi-Layer Artificial Neural Network (MLFF) ANN, k-Nearest Neighbor (k-NN) and Random Forest (RF)).

Table 2. Comparison between algorithms.

Network Traffic Analysis (Binary) (208.629 Instances)							
Classification Accuracy and Performance Metrics							
Classifier	**TA**	**RMSE**	**Precision**	**Recall**	**F-Score**	**ROC Area**	**Time**
SVM	98.01%	0.1309	0.980	0.980	0.980	0.980	273.6 s
MLFF ANN	98.13%	0.1295	0.981	0.981	0.981	0.994	300.2 s
k-NN	96.86%	0.1412	0.970	0.970	0.970	0.970	100.7 s
RF	97.12%	0.1389	0.972	0.971	0.971	0.971	72.2 s
ELM/GRBFK	97.78%	0.1322	0.977	0.977	0.977	0.977	1.9 s

Table 3. Comparison between algorithms.

Demystification of Malware Traffic (Multiclass) (168.501 Instances)							
Classification Accuracy and Performance Metrics							
Classifier	**TA**	**RMSE**	**Precision**	**Recall**	**F-Score**	**ROC Area**	**Time**
SVM	96.63%	0.1509	0.967	0.967	0.968	0.970	101.1 s
MLFF ANN	96.50%	0.1528	0.981	0.981	0.981	0.965	148.3 s
k-NN	94.95%	0.1602	0.970	0.970	0.970	0.950	61.8 s
RF	95.91%	0.1591	0.972	0.971	0.971	0.960	38.7 s
ELM/GRBFK	96.59%	0.1523	0.970	0.980	0.975	0.975	0.91 s

Table 4. Comparison between algorithms.

Encrypted Traffic Analysis (Binary) (166.874 Instances)							
Classification Accuracy and Performance Metrics							
Classifier	**TA**	**RMSE**	**Precision**	**Recall**	**F-Score**	**ROC Area**	**Time**
SVM	98.99%	0.1109	0.989	0.990	0.990	0.990	91.5 s
MLFF ANN	99.12%	0.1086	0.998	0.998	0.998	0.998	116.6 s
k-NN	97.84%	0.1372	0.975	0.975	0.978	0.980	59.2 s
RF	98.96%	0.1107	0.989	0.989	0.989	0.990	40.1 s
ELM/GRBFK	99.20%	0.1056	0.990	0.990	0.990	0.990	0.88 s

Table 5. Comparison between algorithms.

Encrypted Traffic Identification (Multiclass) (214.155 Instances)							
Classification Accuracy and Performance Metrics							
Classifier	**TA**	**RMSE**	**Precision**	**Recall**	**F-Score**	**ROC Area**	**Time**
SVM	90.31%	0.1906	0.905	0.905	0.906	0.950	288.9 s
MLFF ANN	92.67%	0.1811	0.930	0.930	0.928	0.960	312.5 s
k-NN	85.19%	0.2032	0.890	0.890	0.890	0.935	100.9 s
RF	91.56%	0.1800	0.920	0.916	0.916	0.930	78.6 s
ELM/GRBFK	92.65%	0.1813	0.930	0.930	0.930	0.955	2.28 s

Table 6. Comparison between algorithms.

Unencrypted Traffic Identification (Multiclass) (186.541 Instances)							
Classification Accuracy and Performance Metrics							
Classifier	**TA**	**RMSE**	**Precision**	**Recall**	**F-Score**	**ROC Area**	**Time**
SVM	99.92%	0.1003	0.999	0.999	0.999	0.999	119.5 s
MLFF ANN	99.91%	0.1008	0.999	0.999	0.999	0.999	162.9 s
k-NN	98.98%	0.1020	0.989	0.989	0.990	0.995	82.7 s
RF	99.93%	0.1001	0.999	0.999	0.999	0.999	51.5 s
ELM/GRBFK	99.94%	0.1000	0.999	0.999	0.998	0.999	1.84 s

The proposed ELM/GRBFK algorithm seems to have a slightly better performance across all datasets, compared to the other methods but the proposed batch processing approach is hundreds of times faster. Thus, the proposed method is appropriate for big data analytics.

6.2. Streaming Data Classification Performance

The analysis of data streams is a specialized machine learning problem that requires specific metrics to measure the accuracy. The Kappa statistic [51] is the most reliable measure to benchmark the accuracy in streaming data classification. It measures the arrangement between two raters who each classify N items into C equally classes. The explanation of κ is [52]:

$$\kappa = \frac{p_0 - p_e}{1 - p_e} = 1 - \frac{1 - p_0}{1 - p_e} \quad (17)$$

where p_o is the comparative observed arrangement between raters (equal to accuracy) and p_e is the supposed likelihood of chance arrangement, via the observed data to estimate the likelihoods of each observer arbitrarily seeing each class. If the raters are in complete agreement, then $\kappa = 1$. If there is no agreement, $\kappa \approx 0$.

Due to the a temporal dependences in data streams, the Kappa-Temporal statistic was used [51]:

$$\kappa_T = \frac{p - p_{per}}{1 - p_{per}} \tag{18}$$

where p_{per} is the accuracy of the persistent classifier.

The Kappa-Temporal statistic take values within [1, −∞]. If the classifier is seamlessly accurate, then $\kappa_{per} = 1$. In other cases, the $\kappa_{per} = 0$. If $\kappa_{per} < 0$, the reference classifier is performing worse than the baseline classifier [51].

Tables 7–11 presents the results of the scenarios applied on streaming data in this research and the equivalent results from competitive methods (Hoeffding Adaptive Tree (HAT) [52] and primal estimated sub-gradient solver for support vector machine (SPegasos) [53]). The learning estimation used 10,000 instances. Prequential evaluation method was used [54]. The training windows used were 5000 and 1000 instances.

Table 7. Comparison between algorithms.

Network Traffic Analysis				
Performance Metrics				
Classifier	**Window Size 5000**		**Window Size 1000**	
	Kappa Stat	**Kappa Temp Stat**	**Kappa Stat**	**Kappa Temp Stat**
SAM/k-NN	76.90%	77.96%	88.12%	89.64%
HAT	76.87%	77.95%	84.55%	86.19%
SPegasos	76.89%	77.29%	85.02%	87.38%

Table 8. Comparison between algorithms.

Demystification of Malware Traffic				
Performance Metrics				
Classifier	**Window Size 5000**		**Window Size 1000**	
	Kappa Stat	**Kappa Temp Stat**	**Kappa Stat**	**Kappa Temp Stat**
SAM/k-NN	77.02%	78.10%	83.24%	84.36%
HAT	77.06%	78.12%	83.20%	84.01%
SPegasos	77.00%	78.01%	83.02%	84.18%

Table 9. Comparison between algorithms.

Encrypted Traffic Analysis				
Performance Metrics				
Classifier	**Window Size 5000**		**Window Size 1000**	
	Kappa Stat	**Kappa Temp Stat**	**Kappa Stat**	**Kappa Temp Stat**
SAM/k-NN	79.00%	79.94%	86.39%	87.76%
HAT	78.96%	79.91%	82.11%	83.81%
SPegasos	78.98%	79.89%	82.68%	83.52%

Table 10. Comparison between algorithms.

Encrypted Traffic Identification				
Performance Metrics				
Classifier	**Window Size 5000**		**Window Size 1000**	
	Kappa Stat	**Kappa Temp Stat**	**Kappa Stat**	**Kappa Temp Stat**
SAM/k-NN	77.11%	77.54%	84.05%	85.18%
HAT	77.02%	77.35%	80.89%	81.23%
SPegasos	77.03%	77.36%	83.14%	84.16%

Table 11. Comparison between algorithms.

Unencrypted Traffic Identification				
Performance Metrics				
Classifier	**Window Size 5000**		**Window Size 1000**	
	Kappa Stat	**Kappa Temp Stat**	**Kappa Stat**	**Kappa Temp Stat**
SAM/k-NN	76.70%	77.87%	83.91%	85.22%
HAT	76.67%	77.86%	81.08%	81.92%
SPegasos	77.15%	77.95%	82.50%	83.04%

The proposed SAM/k-NN algorithm outperforms the other algorithms by having smaller error rates. More important is that the method produces highly accurate results without non-recurring problems of indeterminate cause because all datasets are better handled. In addition, one of the key advantages that is ascertained from results is kappa reliability, which can be considered as the outcome from the data editing allowing the conservancy of more relevant data for upcoming forecast. The kappa reliability is presented in the following Table 12.

Table 12. Kappa reliability.

Kappa	Reliability
0.00	no reliability
0.1–0.2	minimum
0.21–0.40	little
0.41–0.60	moderate
0.61–0.80	important
≥0.81	maximum

As shown in Table 12, the kappa reliability of the proposed SAM/k-NN algorithm is characterized as "important" in all experiments with windows size 5000 instances and "maximum" with windows size 1000 instances. The results of SAM/k-NN are meaningfully better on the small window because samples of previous instances disappear more quickly with the smaller window and, therefore, less frequently reverse the present concept in the situation of real drift. This proves that the algorithm is highly suitable for applications with streaming process because it is robust to noisy data such as drifting data streams.

7. Conclusions

7.1. Innovation

The most important novelty of λ-NF3 is the proposal of the construction of a NGC2SOC [48] that uses cognitive analytics systems and sophisticated artificial intelligence tools to face real time cyber security incidents with minimal human intervention. In addition, an innovation of the λ-NF3 method is the use of the proposed lambda architecture. This framework uses a versatile and efficient intelligent-driven algorithm for batch processing and a novel evolving learning mechanism for streaming process, to solve an extremely complicated cybersecurity problem.

A basic innovation of this methodology is the combination for the first time in a hybrid machine learning framework the ELM/GRBFK and SAM/k-NN algorithms. The combination offers high learning speed, ease of execution, minimal human involvement and minimum computational power and resources for network traffic analysis, demystification of malware traffic and encrypted traffic identification.

Finally, the datasets, developed after protracted and extensive investigation on the network protocols, work in the lower layers (transport, network and data) and the higher layers (session, presentation and application) of the system. It is important also to note that the dataset is developed after evaluations concerning the restrictions of their characteristic performance of those protocols and the purpose of their normal or abnormal behavior in a real networking environment.

7.2. Discussion

An advanced, dependable and highly effective cybersecurity system, using machine learning principles, is presented in this research paper. It is an appropriate tool for big data applications with many data streams in situations where signature-based approaches are computationally infeasible.

The development of λ-NF3 is based on the Lambda Architecture approach. This architecture can handle enormous quantities of data in real-time using an ideal combination of two machine learning algorithms for batch and data stream. Specifically, it uses batch process to provide complete and accurate views of historical data and real time data stream processing to provide views of new inputs. The final decision comes from the two joined outputs.

The λ-NF3 is an adaptive analytic framework for efficient defense against adversarial attacks and proposed for the NGC2SOC. This intelligence-driven method, from which hopeful outcomes have emerged, creates a reliable advanced application for the tactic of improved cyber security infrastructures. Moreover, this implementation follows a reactive cyber security strategy for dealing with adversarial attacks, as it combines training of two opposite classifiers to detect incoming anomalies and to discard them. Training is done by using sophisticated real datasets that respond to realistic situations. The operating scenarios proposed with the combination of batch and streaming data createabilities for a fully-defined configuration of model parameters and for high-precision classification or correlation. Finally, the application of artificial intelligence on digital recording machines, aiming to recognize adversarial attacks with machine learning, enhances and simplifies the cyber defense and it introduces new perspectives in the management of cyber security policies.

The proposed system was tested and evaluated on real-world datasets of high complexity that emerged after extensive research on network behavior. The remarkable results and the generalization of the system meaningfully support the proposed methodology, although the degree of difficulty and realism that has been added has formed multifactorial questions of exhaustive examination and reproduction.

The evaluation of the proposed method was carried out by thoroughly presenting and quoting the metrics that can determine the classification accuracy of the algorithms. The broad application of the proposed technique, which minimizes the cost of the cyber-attacks, is a prerequisite for the establishment of a NGC2SOC, aiming at the cyber security and protection of organizations and critical infrastructures.

7.3. Future Work

Future enquiry could include the proposed model under a novel structure that would combine semi-supervised approaches and online incremental learning for the identification of hidden patterns between unstructured data types included in network traffic. In addition, λ-NF3 could be improved towards further enhancing the constraints of the algorithm used by the Lambda architecture, so that an even more efficient, more accurate, and faster prediction procedure is achieved. An adapted visualization that can merge into the proposed λ-NF3 would further assist NGC2SOC operators in handling cybersecurity incidents. Multi-format depictions may support a C2 system with advanced reports and representations that enhance the overall decision mechanism. Moreover, it would be significant to study the development of this certain framework by applying lambda architecture in a

parallel and distributed environment such as hadoop. Finally, an additional component that could be considered as a future extension concerns the procedure of λ-NF3 with approaches of self-improvement and meta-learning to fully automate the defense against adversarial attacks.

Author Contributions: Conceptualization, K.D. and N.T.; Investigation, K.D.; Methodology, K.D. and N.T.; Software, K.D. and L.I.; Validation, K.D., N.T., P.K., S.L.S. and L.I.; Formal Analysis, K.D., N.T., P.K., S.L.S. and L.I.; Resources, K.D. and L.I.; Data Curation, K.D., N.T., P.K. and L.I.; Visualization, K.D., N.T. and S.L.S.; Writing—Original Draft Preparation, K.D.; Writing—Review and Editing, K.D., N.T., P.K., S.L.S. and L.I.; and Supervision, N.T.

Funding: This research received no external funding.

Acknowledgments: Nikos Tziritas's work was partly supported by the PIFI International Scholarship, Y75601.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dalvi, N.; Domingos, P.; Sanghai, S.; Verma, D. Adversarial classification. In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), Seattle, WA, USA, 22–25 August 2004; pp. 99–108.
2. Nelson, B.; Barreno, M.; Chi, F.J.; Joseph, A.D.; Rubinstein, B.I.P.; Saini, U.; Sutton, C.; Tygar, J.D.; Xia, K. Exploiting machine learning to subvert your spam filter. In Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET'08), San Francisco, CA, USA, 15 April 2008; pp. 1–9.
3. Fogla, P.; Sharif, M.; Perdisci, R.; Kolesnikov, O.; Lee, W. Polymorphic blending attacks. In Proceedings of the 15th Conference on USENIX Security Symposium (USENIX-SS'06), Vancouver, BC, Canada, 31 July–4 August 2006.
4. Newsome, J.; Karp, B.; Song, D. Paragraph: Thwarting signature learning by training maliciously. In *Recent Advances in Intrusion Detection*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 81–105.
5. Rodrigues, R.N.; Ling, L.L.; Govindaraju, V. Robustness of multimodal biometric fusion methods against spoof attacks. *J. Vis. Lang. Comput.* **2009**, *20*, 169–179.
6. Joseph, A.D.; Laskov, P.; Roli, F.; Tygar, J.D.; Nelson, B. Machine Learning Methods for Computer Security (Dagstuhl Perspectives Workshop 12371). In *Dagstuhl Manifestos*; Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik: Wadern, Germany, 2013; Volume 3, pp. 1–30.
7. Dedić, N.; Stanier, C. *Towards Differentiating Business Intelligence, Big Data, Data Analytics and Knowledge Discovery*; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; p. 285; ISSN 1865-1356; OCLC 909580101.
8. Llopis, S.; Hingant, J.; Pérez, I.; Esteve, M.; Carvajal, F.; Mees, W.; Debatty, T. A comparative analysis of visualisation techniques to achieve cyber situational awareness in the military. In Proceedings of the 2018 International Conference on Military Communications and Information Systems (ICMCIS), Warsaw, Poland, 22–23 May 2018.
9. Demertzis, K.; Iliadis, L. A Hybrid Network Anomaly and Intrusion Detection Approach Based on Evolving Spiking Neural Network Classification. In *E-Democracy 2013: E-Democracy, Security, Privacy and Trust in a Digital World*; Sideridis, A., Kardasiadou, Z., Yialouris, C., Zorkadis, V., Eds.; Communications in Computer and Information Science Series; Springer: Cham, Switzerland, 2014; Volume 441.
10. Demertzis, K.; Iliadis, L. Evolving Computational Intelligence System for Malware Detection. In *Advanced Information Systems Engineering Workshops*; Lecture Notes in Business Information Processing Series; Springer: Cham, Switzerland, 2014; Volume 178, pp. 322–334; doi:10.1007/978-3-319-07869-4_30.
11. Demertzis, K.; Iliadis, L. Bio-Inspired Hybrid Artificial Intelligence Framework for Cyber Security. In *Computation, Cryptography, and Network Security*; Daras, N., Rassias, M., Eds.; Springer: Cham, Switzerland, 2014.
12. Demertzis, K.; Iliadis, L. Bio-Inspired Hybrid Intelligent Method for Detecting Android Malware. In Proceedings of the Advanced Information Systems Engineering Workshops (CAiSE 2014), Limassol, Cyprus, 6–8 November 2014; Iliadis, L., Papazoglou, M., Pohl, K., Eds.; Lecture Notes in Business Information Processing Series; Springer: Cham, Switzerland, 2014; Volume 178.
13. Demertzis, K.; Iliadis, L. Evolving Smart URL Filter in a Zone-based Policy Firewall for Detecting Algorithmically Generated Malicious Domains. In *Statistical Learning and Data Sciences (SLDS 2015)*;

Gammerman, A., Vovk, V., Papadopoulos, H., Eds; Lecture Notes in Computer Science Series; Springer: Cham, Switzerland, 2015; Volume 9047.

14. Demertzis, K.; Iliadis, L. SAME: An Intelligent Anti-Malware Extension for Android ART Virtual Machine. In *Computational Collective Intelligence*; Núñez, M., Nguyen, N., Camacho, D., Trawiński, B., Eds.; Lecture Notes in Computer Science Series; Springer: Cham, Switzerland, 2015; Volume 9330.
15. Demertzis, K.; Iliadis, L. Computational Intelligence Anti-Malware Framework for Android OS. *Vietnam. J. Comput. Sci.* **2017**, *4*, 245–259, doi:10.1007/s40595-017-0095-3.
16. Demertzis, K.; Iliadis, L. Ladon: A Cyber-Threat Bio-Inspired Intelligence Management System. *J. Appl. Math. Bioinform.* **2016**, *6*, 45–64.
17. Demertzis, K.; Iliadis, L.S.; Anezakis, V.-D. An innovative soft computing system for smart energy grids cybersecurity. *Adv. Build. Energy Res.* **2018**, *12*, 3–24.
18. Scandariato, R.; Walden, J. Predicting vulnerable classes in an android application. In Proceedings of the 4th International Workshop on Security Measurements and Metrics, Lund, Sweden, 21 September 2012.
19. Shabtai, A.; Fledel, Y.; Elovici, Y. Automated static code analysis for classifying android applications using machine learning. In Proceedings of the 2010 International Conference on Computational Intelligence and Security, Nanning, China, 11–14 December 2010; pp. 329–333.
20. Chin, E.; Felt, A.; Greenwood, K.; Wagner, D. Analyzing inter-application communication in android. In Proceedings of the 9th Conference on Mobile Systems, Applications, and Services, Bethesda, MD, USA, 28 June–1 July 2011; pp. 239–252.
21. Burguera, I.; Zurutuza, U.; Nadjm-Tehrani, S. Crowdroid: Behavior-based malware detection system for android. In Proceedings of the1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, Chicago, IL, USA, 17 October 2011; pp. 15–26.
22. Glodek, W.; Harang, R.R. Permissions-based Detection and Analysis of Mobile Malware Using Random Decision Forests. In Proceedings of the IEEE Military Communications Conference, San Diego, CA, USA, 18–20 November 2013.
23. Zhang, J.; Chen, C.; Xiang, Y.; Zhou, W.; Vasilakos, A.V. An effective network traffic classification method with unknown flow detection. *IEEE Trans. Netw. Serv. Manag.* **2013**, *10*, 133–147.
24. Joseph, G.; Nagaraja, S. On the reliability of network measurement techniques used for malware traffic analysis. In Proceedings of the Cambridge International Workshop on Security Protocols, *Cambridge*, UK, 19–21 March 2014; pp. 321–333.
25. Hsu, C.-H.; Huang, C.-Y.; Chen, K.-T. Fast-flux bot detection in real time. In Proceedings of the 13th International Conference on Recent Advances in Intrusion Detection (Ser. RAID'10), Ottawa, ON, Canada, 15–17 September 2010.
26. Haffner, P.; Sen, S.; Spatscheck, O.; Wang, D. ACAS: Auto-mated Construction of Application Signatures. In Proceedings of the 2005 ACM SIGCOMM Workshop on Mining Network Data, Philadelphia, PA, USA, 22–26 August 2005; pp. 197–202.
27. Holz, T.; Gorecki, C.; Rieck, K.; Freiling, F. Measuring and detecting fast-flux service networks. In Proceedings of the Network & Distributed System Security Symposium (NDSS'08), San Diego, California, USA, 10 - 13 February 2008.
28. Almubayed, A.; Hadi, A.; Atoum, J. A Model for Detecting Tor Encrypted Traffic using Supervised Machine Learning. *Int. J. Comput. Netw. Inf. Secur.* **2015**, *7*, 10–23.
29. HoseinyFarahabady, M.; Taheri, J.; Tari, Z.; Zomaya, A.Y. A Dynamic Resource Controller for a Lambda Architecture. In Proceedings of the 2017 46th International Conference on Parallel Processing (ICPP), Bristol, UK, 14–17 August 2017; pp. 332–341; doi:10.1109/ICPP.2017.42.
30. Suthakar, U.; Magnoni, L.; Smith, D.R.; Khan, A. Optimised lambda architecture for monitoring WLCG using spark and spark streaming. In Proceedings of the 2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD), Strasbourg, France, 29 October–6 November 2016; pp. 1–2, doi:10.1109/NSSMIC.2016.8069637.
31. Kiran, M.; Murphy, P.; Monga, I.; Dugan, J.; Baveja, S.S. Lambda architecture for cost-effective batch and speed big data processing. In Proceedings of the 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, USA, 29 October–1 November 2015; pp. 2785–2792; doi:10.1109/BigData.2015.7364082.
32. Yamato, Y.; Kumazaki, H.; Fukumoto, Y. Proposal of Lambda Architecture Adoption for Real Time Predictive Maintenance. In Proceedings of the 2016 Fourth International Symposium on Computing and

Networking (CANDAR), Hiroshima, Japan, 22–25 November 2016; pp. 713–715; doi:10.1109/CANDAR.2016.0130.
33. Yong, S.Z.; Foo, M.Q.; Frazzoli, E. Robust and resilient estimation for Cyber-Physical Systems under adversarial attacks. In Proceedings of the 2016 American Control Conference (ACC), Boston, MA, USA, 6–8 July 2016; pp. 308–315; doi:10.1109/ACC.2016.7524933.
34. Chong, M.S.; Wakaiki, M.; Hespanha, J.P. Observability of linear systems under adversarial attacks. In Proceedings of the 2015 American Control Conference (ACC), Chicago, IL, USA, 1–3 July 2015; pp. 2439–2444; doi:10.1109/ACC.2015.7171098.
35. Chen, L.; Ye, Y.; Bourlai, T. Adversarial Machine Learning in Malware Detection: Arms Race between Evasion Attack and Defense. In Proceedings of the 2017 European Intelligence and Security Informatics Conference (EISIC), Athens, Greece, 11–13 September 2017; pp. 99–106; doi:10.1109/EISIC.2017.21.
36. Wang, W.; Zhang, X.; Shi, W.; Lian, S.; Feng, D. Network traffic monitoring, analysis and anomaly detection [Guest Editorial]. *IEEE Netw.* **2011**, *25*, 6–7, doi:10.1109/MNET.2011.5772054.
37. Xu, C.; Chen, S.; Su, J.; Yiu, S.M.; Hui, L.C.K. A Survey on Regular Expression Matching for Deep Packet Inspection: Applications, Algorithms, and Hardware Platforms. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 2991–3029, doi:10.1109/COMST.2016.2566669.
38. Zhang, H.; Papadopoulos, C.; Massey, D. Detecting encrypted botnet traffic. In Proceedings of the 2013 IEEE Conference on Computer Communications Workshops, Turin, Italy, 14–19 April 2013; pp. 3453–1358; doi:10.1109/INFCOM.2013.6567180.
39. Aretz, K.; Bartram, S.M.; Pope, P.F. Asymmetric Loss Functions and the Rationality of Expected Stock Returns. *Int. J. Forecast.* **2011**, *27*, 413–437, doi:10.1016/j.ijforecast.2009.10.008.
40. Kushner, H.J.; Yin, G.G. *Stochastic Approximation and Recursive Algorithms and Applications*, 2nd ed.; Springer: New York, NY, USA, 2003; ISBN 0-387-00894-2.
41. Cambria, E.; Huang, G.B.; Kasun, L.L.C.; Zhou, H.; Vong, C.M.; Lin, J.; Yin, J.; Cai, J.; Liu, Q.; Li, K.; et al. Extreme learning machines [trends & controversies]. *IEEE Intell. Syst.* **2013**, *28*, 30–59.
42. Huang, G.-B. An Insight into Extreme Learning Machines: Random Neurons, Random Features and Kernels. *Cogn. Comput.* **2014**, *6*, 376–390, doi:10.1007/s12559-014-9255-2.
43. Huang, G.-B. What are Extreme Learning Machines? Filling the Gap between Frank Rosenblatt's Dream and John von Neumann's Puzzle. *Cogn. Comput.* **2015**, *7*, 263–278, doi:10.1007/s12559-015-9333-0.
44. Losing, V.; Hammer, B.; Wersing, H. KNN Classifier with Self Adjusting Memory for Heterogeneous Concept Drift. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM), Barcelona, Spain, 12–15 December 2016; pp. 291–300, doi:10.1109/ICDM.2016.0040.
45. Haining, W.; Danlu, Z.; Kang, G.S. Detecting SYN flooding attacks. In Proceedings of the INFOCOM 2002—Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, New York, NY, USA, 23–27 June 2002; Volume 3, pp. 1530–1539.
46. Arndt, D.J.; Zincir-Heywood, A.N. A Comparison of Three Machine Learning Techniques for Encrypted Network Traffic Analysis. In Proceedings of the 2011 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), Paris, France, 11–15 April 2011; pp. 107–114.
47. contagiodump. Available online: http://contagiodump.blogspot.gr/ (accessed on 27 September 2018).
48. Demertzis, K.; Kikiras, P.; Tziritas, N.; Sanchez, S.L.; Iliadis, L. The Next Generation Cognitive Security Operations Center: Network Flow Forensics Using Cybersecurity Intelligence. *Big Data Cogn. Comput.* **2018**, *2*, 35.
49. Mao, J.; Jain, A.K.; Duin, P.W. Statistical pattern recognition: A review. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 4–37.
50. Fawcett, T. An introduction to ROC analysis. *Pattern Recognit. Lett.* **2006**, *27*, 861–874, doi:10.1016/j.patrec.2005.10.010.
51. Žliobaitė, I.; Bifet, A.; Read, J.; Pfahringer, B.; Holmes, G. Evaluation methods and decision theory for classification of streaming data with temporal dependence. *Mach. Learn.* **2015**, *98*, 455–482, doi:10.1007/s10994-014-5441-4.
52. Corrêa, D.G.; Enembreck, F.; Silla, C.N. An investigation of the hoeffding adaptive tree for the problem of network intrusion detection. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 4065–4072; doi:10.1109/IJCNN.2017.7966369.
53. Shalev-Shwartz, S.; Singer, Y.; Srebro, N.; Cotter, A. Pegasos: Primal estimated sub-gradient solver for SVM. *Math. Program.* **2011**, *127*, 3–30, doi:10.1007/s10107-010-0420-4.

54. Vinagre, J.; Jorge, A.M.; Gama, J. Evaluation of recommender systems in streaming environments. In Proceedings of the Workshop on 'Recommender Systems Evaluation: Dimensions and Design' (REDD 2014), Silicon Valley, CA, USA, 10 October 2014; doi:10.13140/2.1.4381.5367.