applied sciences

MDPI

Article

Detecting Cryptojacking Web Threats: An Approach with Autoencoders and Deep Dense Neural Networks

Aldo Hernandez-Suarez [1], Gabriel Sanchez-Perez [1], Linda K. Toscano-Medina [1], Jesus Olivares-Mercado [1,*], Jose Portillo-Portilo [1], Juan-Gerardo Avalos [1] and Luis Javier García Villalba [2,*]

[1] Instituto Politecnico Nacional, ESIME Culhuacan, Mexico City 04440, Mexico; alhernandezsu@ipn.mx (A.H.-S.); gasanchezp@ipn.mx (G.S.-P.); ltoscano@ipn.mx (L.K.T.-M.); jportillop@ipn.mx (J.P.-P.); javaloso@ipn.mx (J.-G.A.)

[2] Group of Analysis, Security and Systems (GASS), Department of Software Engineering and Artificial Intelligence (DISIA), Faculty of Computer Science and Engineering, Office 431, Universidad Complutense de Madrid (UCM), 28040 Madrid, Spain

* Correspondence: jolivares@ipn.mx (J.O.-M.); javierv@ucm.es (L.J.G.V.); Tel.: +52-555624200 (J.O.-M.); +34-913947638 (L.J.G.V.)

Abstract: With the growing popularity of cryptocurrencies, which are an important part of day-to-day transactions over the Internet, the interest in being part of the so-called cryptomining service has attracted the attention of investors who wish to quickly earn profits by computing powerful transactional records towards the blockchain network. Since most users cannot afford the cost of specialized or standardized hardware for mining purposes, new techniques have been developed to make the latter easier, minimizing the computational cost required. Developers of large cryptocurrency houses have made available executable binaries and mainly browser-side scripts in order to authoritatively tap into users' collective resources and effectively complete the calculation of puzzles to complete a proof of work. However, malicious actors have taken advantage of this capability to insert malicious scripts and illegally mine data without the user's knowledge. This cyber-attack, also known as cryptojacking, is stealthy and difficult to analyze, whereby, solutions based on anti-malware extensions, blocklists, JavaScript disabling, among others, are not sufficient for accurate detection, creating a gap in multi-layer security mechanisms. Although in the state-of-the-art there are alternative solutions, mainly using machine learning techniques, one of the important issues to be solved is still the correct characterization of network and host samples, in the face of the increasing escalation of new tampering or obfuscation techniques. This paper develops a method that performs a fingerprinting technique to detect possible malicious sites, which are then characterized by an autoencoding algorithm that preserves the best information of the infection traces, thus, maximizing the classification power by means of a deep dense neural network.

Keywords: cryptomining; machine learning; antoencoders

Citation: Hernandez-Suarez, A.; Sanchez-Perez, G.; Toscano-Medina, L.K.; Olivares-Mercado, J.; Portillo-Portilo, J.; Avalos, J.-G.; García Villalba, L.J. Detecting Cryptojacking Web Threats: An Approach with Autoencoders and Deep Dense Neural Networks. *Appl. Sci.* **2022**, *12*, 3234. https://doi.org/10.3390/app12073234

Academic Editor: Agostino Forestiero

Received: 24 January 2022
Accepted: 16 March 2022
Published: 22 March 2022

1. Introduction

The constant technological evolution, which, thanks to the ubiquity of the Internet and the services offered by cloud computing, allows a large part of human activities to be developed in an online and digital way, through the widespread use of electronic devices and computers. The design of new solutions offered as scalable and on-demand systems has connected society in fields such as education, health, privacy and security, culture, personal development, and, to a large extent, e-commerce. This has served to transform technical information, previously understood only by experts, into easy-to-understand ontologies for the average end-user, to operate their technology needs. As a result, there is a big demand for access to this new era of easy digitization of payments, benefiting monetary transactions that are now better comprised and can be carried out instantaneously, overcoming currency or regional barriers [1,2].

The new era of digital finance has been gradually replacing the physical form of money provision with an electronic one, which, in turn, involves a better level of independence, but has consequences for the privacy and confidentiality of payments.

Indeed, the security and anonymity offered by VPNs, together with sophisticated authentication mechanisms and multi-factor add-ons [3] have gained the trust of the common user, but undoubtedly one of the forms of electronic money that have been adopted in a considerable way are cryptocurrencies, which touch on the advantages mentioned before and also the freedom of banking intermediaries: digital cash that is handled on a peer-to-peer (P2P) basis. The appeal of cryptocurrencies is that they offer a distributed transactional model, using cryptographic techniques and a puzzle-solving stage to address the risks of counterfeiting and double-spending, also known as anti-tampering resistance. Although a degree of confidentiality and integrity is implied, cryptocurrencies are not backed by a trusted third party; however, this has not limited their commercial use, which has increased since their creation in 2009, becoming a newly popular method for online payment transactions and investment [4]. In fact, the interest in cryptocurrencies has spread to such an extent, that it is a matter of selecting the desired cryptocurrency house, reading the documentation provided, using a recommended application and then start monitoring trends, mining blocks, or simply making the desired monetary transactions [5]. The cryptocurrency boom has been presented as a consumer commodity, even used as part of marketing campaigns. One of the most convincing cases is the one explained in [6], in which airlines suggested the adoption of the advertising of large cryptocurrency houses as a promotional medium on the Internet and the generation of more popularity ratings, within them.

Many cryptocurrencies are decentralized networks based on blockchain technologies, i.e., distributed operations enforced by a network of computers, involving an arduous computational process to cryptomine new blocks towards the blockchain network. This procedure, namely, the proof-of-Work (PoW), is a pool of consensus, aimed to solve a puzzle that avoids cheating the system by solving expensive and difficult cryptographic calculations, to eventually gain the binding of a new block, restricted to a hashing rate. The latter is performed thanks to the distributive capabilities of various device networks, where different participants provide, in an individual manner, processing capacity in terms of specialized hardware, GPUs, FPGAs, or ASICs, which can efficiently protect the transaction, given the algorithmic power, and the resistance to fault attacks [7]. Finally, if the block is successfully constructed, users are rewarded with a certain amount of the cryptocurrency, which is undoubtedly a lucrative business for digital money enthusiasts.

The growing phenomenon of investing in crypto-processing has led developers to implement more flexible ways for solving PoWs. For example, Monero, the once-popular cryptocurrency brokerage company, designed an algorithm called CryptoNight, which offered new opportunities to process calculations over standard CPU-oriented systems. With this in mind, several POW systems opted to take advantage of the large-scale communication capabilities of the Internet, thus giving birth to web-based cryptomining, which has as its main advantage, the distributed use of computational resources. With the consent of a large number of users, the scripts are executed from the carrier site, through the browser, which binds the puzzle fraction to be solved to the CPU, adding it as another node in the transactional network [8].

In the same way, the market demand has increased the commercial value of cryptocurrencies, with Bitcoin, Ethereium, Litecoin, Dogecoin, and Riple as main exchanges, all of which offer attractive ways to join the PoW network, via system software, shareware, or web scripting [9]. An interesting example is what happened during the early days of the COVID-19 pandemic, which according to [10], special attention was paid to cryptocurrency investment as an alternative means of profit, with Bitcoin being the dominant currency in the market. Given this ease of joining the network of crypto-processing competitors, malicious actors have also been attracted to make money illicitly, leveraging classic attack vectors such as malware, botnets, or by simply taking advantage from already discovered

security breaches, to embed malicious mining code over IT infrastructure, targeting numberless third-party computing services; therefore, the rise of illegal mining has alarmed governments and security providers, assessing illegitimate cryptocurrency as a significant threat over the Internet, affecting numerous visitors who fall prey to the lure of infected sites. Based on a CTA (Cyber Threat Alliance) white paper, the growth of counterfeited cryptocurrency mining has intensified since the last months of 2017 and shows a rapid development with no signs of slowing down [11].

Although infections by cryptocurrency-related malware can be found on malicious links in email messages and attachments, applications with masqueraded activity and hijacked browser extensions, the last, remains as the main and most lucrative way of web-based-mining scripting, namely, web-based cryptojacking; which according to [12] contains stealthy attack vectors to make use of several computational resources without any restriction. This is a great advantage on the attacker's side as it only requires the visitor to stay long enough on the website to proceed with the execution of the malicious code. In a point of fact, this implies a critical security point, since this form of cryptojacking does not require the download of executables, binaries, or adware, but as a primary tool, a common web browser, which is a native component of numerous operating systems from everyday devices: personal computers, workstations, smart-phones, and IoT gadgets.

As an example of malware persistence, between the years 2017 and 2018, Coinhive, one of the most expanded malicious platforms for the development of cryptojacking-oriented scripts, quickly defaced a significant number of websites, via server-side content injection, where more than 45,000,000 illicit transactions were recorded in that period [13]. Figure 1 describes different platforms, which are used as cryptocurrency malware or web-based cryptojacking.

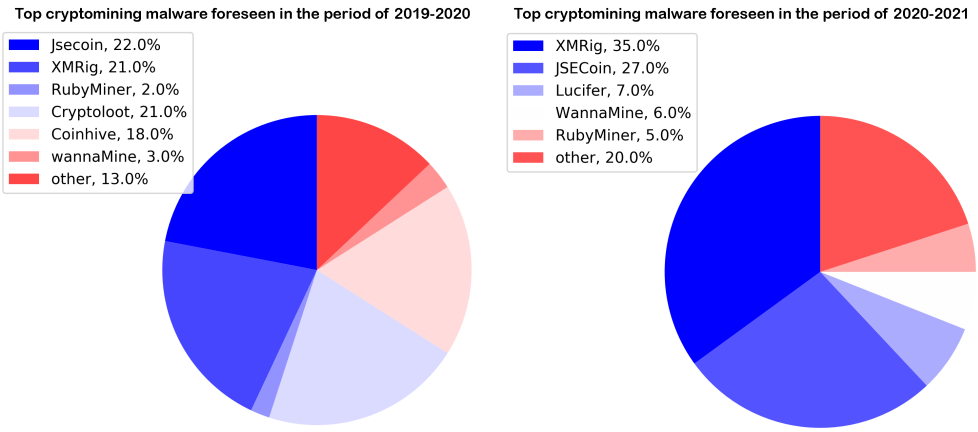


Figure 1. On the authority of ENISA and the Internet Organised Crime Threat Assessment (IOCTA), a list of illegal cryptocurrency abuse platforms is presented, as part of the most frequent and harmful threats since 2019 [14,15].

Web-based cryptojacking is a threat that seems difficult to fade, according to a report by the European Union Agency For Cybersecurity (ENISA), between January 2019 and April 2020 were detected 64.1 million sites with some kind of unlawful cryptocurrency activities, with novel and more sophisticated attack vectors. In addition, the ENISA confirmed that 65% of cryptocurrency transactions are subject to weak or leaky know-your-customer-processes, leaving a loophole in the illegal exploitation of IT resources for cryptomining purposes. This was increased severely during the first months of 2021, where a significant increase in the illicit cryptomining market was reported, and although the trend is against infected files, web-based propagation is still striking. An example of this was zero-day attacks on unpatched Microsoft Exchange Servers, where the first payloads had traces of web-based cryptomining code, with new tactics of concealment.

Even though the shutdown of several classic web-based cryptocurrency sites is well known, the attention of malicious actors has shifted to cloud infrastructures, exploiting APIs and containers (e.g., Docker and Kubernetes) to install illegal cryptocurrency images. In any case, the strength of cryptojacking strains is coupled with the handiness of coding, thanks to the incorporation of technologies such as WebSAssembly, WebWorkers, and Websockets, which simplify the execution of high-performance multi-platform scripts, intensifying the number of potential victims [16].

Being a threat that encompasses a high proportion of risk, due to its high level of profitability, it is therefore necessary to understand the current landscape, in order to gather the necessary elements for timely detection. Classical reverse engineering scopes and those based on anomaly modeling have not been sufficient to project the threat. Beforehand it can be considered that the user's activities are the primary task in information security events, the same that allow to trace failures, intrusions, bad functionality, and in general any point that is considered surface and vector of attack, where it interacts [17]. An important point in this type of research related to a malicious agent that spreads on different surfaces is to understand the taxonomy of the anomaly, therefore, in [18] it is proposed that it is necessary for security systems to reason the change in the normal activity flows and to be able to discriminate similar/non-similar objects.

This manuscript takes the above references as a key starting point and is intended to contribute to solve the problem of web-based cryptojacking with a novel form of latent representation of malicious traces, which are trained by a deep learning algorithm that maximizes the capabilities of sensing, from different flanks, specifically network, and host characteristics. The results discussed later in Section 6 show that by means of the strategy proposed in this paper it is possible to achieve a Precision of 99.41%, a Recall with a score of 99.11%, and an F1 Score of 99.26%, with a time no longer than a minute and a half for the generation of the final model. Thus, this methodology can be quickly adapted to different threat properties and attack the problem of web-based cryptojacking.

The remainder of the manuscript is organized as follows. Section 2 explains the motivation and background that gave rise to this research. Section 3 describes the phenomenon of cryptomining, the threats of web-based cryptojacking, and its prevalence. Section 4 discusses the most employed detection methods, the scope of the algorithms, and the related work through the incorporation of ML techniques. In Section 5, the AE and DDNN framework is extensively described. Section 6 describes the results obtained, their discussion, and improvements presented against other state-of-the-art works. Finally, Section 7 concludes this work.

2. Background and Motivation

Like any other security threat, cryptojacking has been subjected to different solutions for its mitigation and eradication. As a malware where the cryptographic element is present, there is a relationship between new strands, such as crypto-ransomware, where symmetric cipher blocks can be distinguished during the exploitation stage and heuristic flagging from different antivirus vendors [19].

Beyond detecting such malware merely by cryptographic elements, the first scans presented in the state-of-the-art have distinguished the inherent properties of the executable as a starting point for examinations by means of the well-known static (SA) and dynamic (DA) analyses. The last-mentioned can provide preliminary results that can be used for triaging and exploratory data analysis, which can reveal important points in end-to-end network traffic forensics, source code auditing, CPU stress operations, and functional monitoring of infected sites, within user interactions [16]. It is well known that SA and DA are the first and primordial stage for discovering and tracing a malicious agent in a chain of custody, however, these practices, lack of automation properties, and therefore any kind of knowledge-base to further learn from significant patterns, beyond what can be humanly configured, interpreted, and managed using a wide repertoire of security tools. Therefore, the outputs of SA and DA cannot be used to construct simple detection signatures, nor

to be released to work as a single security mechanism, instead, the resulting data can be properly characterized and tuned to serve as input for machine learning ML algorithms, capable of discovering important underlying information [20].

The literature for detection, classification, and clustering of samples for timely detection of web-based cryptomining threats, especially for observations of different varieties of cryptojacking, comprises a broad spectrum of supervised learning (SL) [21], unsupervised learning (UL) [22], and deep learning (DL) algorithms [23], where an emphasis is placed on the true positive (TP) radius enhancement. It is worth mentioning, that the above-named fields presented interesting findings with more than 90% of accuracy for cryptojacking classification. Despite their theoretical effectiveness, there is no consensus on what set of features can represent the threat as a whole, given the attack surfaces where it unfolds. The first surface resides on the LAN and WAN (Internet) networks since it is the point where traffic connections are made between the attacker's PoW and the victim executing the malicious script. The second surface involves the events that occur on the client (host) side when the malicious script is running. In this part, it is possible to log the activities from the operating system related to the consumption of resources from the browser during the time it is hijacked. The purpose of outlining the above is simple: by adding features from the network and host side, the generalization of the cryptojacking problem can be increased, thus expanding the likelihood of successful feature extraction, selection, and learning process.

In agreement with [24], when incorporating ML solutions in the cybersecurity domain, it is important to take into account a holistic view of the event occurring on the network and host fronts; therefore, the algorithm proposal is not enough, we also require the acquisition, processing, and proper characterization of the observed and fused data. Cryptojacking does not operate in a traditional way compared to other types of malware, its nature is a pivot attack, where the browser concentrates the payloads and acts as an intermediary between the remote site and the user, without a direct infection. So then, it is important to meet the criteria for an optimal representation of the observed samples, from various edges. A great challenge can be encountered when capturing a pivoted threat: samples from different but interlinked origins can lead to a large occurrence of outliers due to dissimilar measurements, sizes, and types of data, incrementing the chances of high dispersion, as some records could only be observed under the presence of well-configured sandboxed cryptojacking instance.

As described in Section 3, there are two important gaps in the state-of-the-art regarding cryptojacking detection: 1. In the first instance, it has already been proven that traditional scopes are outdated and not sufficient, therefore SA and DA techniques can serve only as a basis for congregating sample-features 2. The source and selection of the features, either at the network or host level, may be merged to achieve a desired level of detection, but not to be considered separately. This raises an important research question:

Research question: How to optimize web-based cryptojacking detection by a procedure that takes the best of network and host features and merges them?

This work concentrates on a new form of cryptojacking feature representation technique, recently adopted for ML-oriented cybersecurity scenarios, called autencoding (AE) [25], which is a specific structure of neural network (NN) focused on compressively capturing the latent information from unstructured data, in virtue of dissimulating the effects of noise, high dimensionality, and non-linearity. Then, the outputs are subjected to a deep dense neural network (DDNN) [26], a type of deep neural network, which has demonstrated nice effectiveness in the treatment of a vast number of malware-related scenarios. As a sign of a successful performance, the resulting model is compared with different methodologies presented in the state-of-the-art for cryptojacking detection.

3. Cryptomining, Cryptojacking, and Web-Based Mining Threats

As mentioned in Section 1, legal cryptomining is an attractive activity in order to earn a profit by mining new blocks to the blockchain-based currency network, where the puzzle-solving process is the most lucrative component, within a network of competitors. At its most fundamental level, cryptocurrency mining is simply the use or reuse of any system resource to increase the performance of complex mathematical calculations over a PoW, usually involving a hashing-type solution. These calculations maintain and validate the transaction history of the distributed ledger, generally known as the blockchain-net. As the cryptocurrency network grows, the calculations become more difficult, making individual machines less likely to solve equations with their limited processing power. Hence, it is common for specialized hardware components, better known as ASICs, to be used in conjunction with other workstations to serve as cryptomining farms. Consequently, alternatives have been sought when sufficient processing capacity is not available, where collaborative works (P2P nodes) are a way to pool the distributed power of single devices and solve more expensive computations [27]. The latter is known as pool mining, a technique aimed to help parallelized mining procedures and successfully create new blocks to obtain an expected reward.

The phenomenon of distributed lending of resources has led to a fierce career within the mining chain since the final dividend is proportional to the hashing power, i.e., the performance of each individual miner, which depends directly on the speed to verify the data through hashing algorithms: the faster the performance, the more cryptocurrency the miner will receive [28]. The arduous struggle to complete mining tasks, especially those implemented by Bitcoin, proved that the exponential volume for power consumption and the difficulty of the PoW challenges, require extensive configurations found in ASICs such as FPGAs or GPUs, and therefore, an unattractive high-cost investment in the face of the growing demand for faster hashing operations. In consequence, other cryptocurrencies such as Monero decided to turn the well-known P2P collaborative network into software alternatives, specifically, scripts that can be run inside a web browser, proficient in exploiting average CPU architectures [29].

This fashioned form of rapid access to cryptomining has generated a significant enrichment within the stakeholders, incorporating a new kind of informal economy, since miners do not need to have deep knowledge of IT, only the ability to build and operate a command base with sufficient computational power, to rapidly construct emerging blocks [30]. These attractive financial movements have engaged the attention of malicious actors, who take advantage of the distribution of resources, thus abusing personal devices and ultimately, profiting without funding or adequate infrastructure. Illegal cryptomining is a stealthy cybercrime, the victim usually is not aware of being hijacked, leaving few indicators of compromise, in such a way to be difficult to detect at a first glance [31]. This class of threats, attacks, and electronic crimes are included as cryptojacking, since they share the same goal: an unauthorized way of cryptomining, which is carried out by stealing resources of one or several victims, to exfiltrate specialized hashes towards the PoW endpoints.

Because of the diverse range of attack vectors, cryptojacking can be set up on mobile devices, software, binaries, network appliances, compromised third-party libraries, botnets, IoT instruments, and server-side applications [13,32]. With those residing on the web being the most prevalent, given the simplicity of inserting a few lines of code into a website and thus forcing the execution of the malicious payload [33] against the visitor's CPU. For its part, the now-defunct Monero, was one of the pioneers in developing web-based cryptomining with ASIC resistance, allowing the incorporation of scripts compatible with common desktop processors and a wide range of operating systems.

The sudden use of Monero API intermediary programs and the extensibility/reusability of the code allowed the accelerated expansion of sites with embedded mining scripts, which in most cases did not require user consent, as well as an authorization note to warn that the site contains cryptomining code. According to [34], due to a large number of visits in a short time, streaming sites, online games, pornographic sites, download shortener redirection sites, and e-commerce platforms remain the main target for attackers to compromise the dynamics of the web page by inserting arbitrary code; however, the constant search for more feasible and integrated (all-in-one) means to disseminate the Monero code led to the development of frameworks such as Coinhive, capable of capturing 30% of new block transactions on the black market, and designed with anonymity options such as no traceability and no perceptibility at a browser level. Although Monero started with a benign purpose, it gradually grew as a hazardous strain that quickly spread over vulnerable websites, being almost 81% of the adware threats formally detected in [35].

Despite the fact that Monero closed its operations in 2019, cyberattacks related to browser-based cryptomining are still present and with regular intervals of an uprising. An example of which are recent malicious scripts such as Cryptoloot, JSECoin, Miner, XMROmine, and deepMinerAnonymous, that according to statistics presented in [36], as of 2020 were found within 66,000 domains with 1.4 million active URLs.

To conclude this section, it is important to mention that the speculative price of cryptocurrencies, as it is a flourishing phenomenon, will continue to attract cybercriminals to attempt the disruption of web applications, at the expense of innumerable vulnerabilities, poor source code auditing, lax security configurations, and intrusion detection/prevention systems or antivirus with inadequate signatures.

4. Detecting Browser-Based Cryptojacking and Related Works

As many other viruses, malware, and suspicious objects, the first line of defense against cryptojacking is through antiviruses and web browser extensions, anti-malware software, firewalls, intrusion detection (IDS), or intrusion prevention systems (IPS), which depending on their structure, share as a common point: an up-to-date detection signature capable of performing a match and flag procedure [37]. Nevertheless, it is important to note that such solutions must differentiate between those contexts where the user expressly agrees to the execution of cryptomining scripts and those who do not, making this, a difficult task to accomplish, since a signature-based appliance works by mapping sequences of bits towards a targeted object and presents the result in terms of similarity, which is, therefore, not an expert system, but an effective one for detecting already observed samples. On the other hand, cryptojacking detection can also be categorized into host-based defenses, including primarily, the so-called resource consumption analysis (RCA) [31], which use the threads generated by the web browser to detect if the use of computational resources on the client side is being abused by comparing with previously established thresholds.

In addition, with SA, a well-known technique for evaluating syntactically and semantically patterns in source code, bytecodes, opcodes, and configurations of an executable or binary, malicious cryptomining scripts [38] can be partially identified and blocked from execution. An example of SA, is the proposed NoCoin blacklisting of the Opera browser, which is used as an extension, to block requests from malicious net ranges [39]. In [29], a hybrid scope of SA and DA over time series is presented, where malware feeds and OSINT (Open Source Intelligence) techniques are merged to differentiate between different classes of executable binaries: illegal and benign cryptomining. To accomplish the workflow, data from URLs and public domains were scraped to monitor wallets and mining pools and determine the number of records generated per block. With this, the hashing ratio can be computed to provide sufficient values to conclude if the binary is considered a cryptojacking attack scenario as part of a botnet or produced by a single attacker.

Further on, in [40] a detection scenario is presented by identifying PoWs signatures and a profile detector, based on the calculation of workloads from several websites appended on Alexa's Top Domain list [41]. It is concluded that by means of a considerable number of heavy hash rates, measured by a stack structure-based profiler (SSBP), it is possible to verify if a signature corresponds to an abnormal cumulative processing time, compared to regular loads of benign sites. In the same sense, in [8] a reconnaissance system was developed to find different types of cryptomining scripts distributed in popular search categories such as: technology and communications, games, dynamic sites, business, pornography, and digital marketing. The system works by determining the origin of a malicious transaction and mapping the link between the PoW and the Merkle tree used to mine the block. With this, it is possible to discern between an indirect command execution, the size of the mining network and the destination of the forged request.

In the last SA scope, in [42], a system titled MANiC: Multi-step Assessment for Cryptominers was implemented with the ability to parse different sites with possible illegal mining or cryptojacking, using a web crawling method composed with several regular expressions related to already flagged code syntax. The authors demonstrate that MANiC is able to recognize more than fourteen crypto-malware families.

Among the works using DA, in [43] a controlled sandbox environment is fed with samples of different malicious scripts, then, monitoring techniques are used to classify the irregular behavior of crypto-malware compared to benign software, in terms of sequences from the processing opcode flows and win32 api functions. In the same vein, the authors in [44] proposed to identify connections between the web browser and dynamic executions of programs running in JavaSript and WebAssembly (WASM). It is argued that due to the sophistication of obfuscation methods in the script code, detection by SA or DA may be difficult. Instead, if semantic signatures are defined, the detection will be more accurate as it pinpoints on portions of crafted WASM code that cannot be hidden.

Although advances in cryptojacking detection are important at different layers and levels via SA, DA, or both, it is mentioned in [45] that most of the physical and logical devices that offer solutions based on pre-built signatures have proven not to be as effective due to a high false positive rate, which is a major disadvantage for unveiling unfamiliar samples.

Even so, in [46] it is discussed that SA and DA processes executed individually are not entirely useful for building an effective detection system, but rather, they serve as initial steps for building datasets that will lead to the design of intelligent sensors that can figure deeper patterns. This has gradually led to more robust detection systems, which as a nucleus use artificial intelligence (AI) algorithms to assemble and enhance the capabilities of traditional solutions.

One of the first works to employ ML for cryptojacking detection is [31], where a complex feature analysis based on SA and web content, specifically JavaScript source code, is applied. On the content side, it is shown that web-based cryptocurrency mining is a widespread threat that can be quickly camouflaged within other scripts and tags, mostly due to security holes in request filtering, allowing arbitrary injection of malicious objects. The authors inferred that a large part of the samples found belonged to Coinhive, where an extensive demand for counterfeited calculations around Monero's PoW points was revealed, being the main source of cryptojacking-related traffic. The authors estimated that through the application of algorithms called cyclomatic complexity (CC) and cyclomatic complexity density (CCD), it is possible to formally screen script execution flows, which are assessed by a non-supervised learning (NSL) process, aimed to differentiate groups of benign and malignant sites.

On the other hand, in [47], BMDetector was presented as a framework that locates some peculiarities in the execution of client-side codes programmed in JavaScript: first, the heap snapshot is monitored as a process tree between the document object model (DOM) nodes and the malicious script triggers, in order to cover samples that could be obfuscated. Then, a representative set of memory stack-related features is presented and evaluated by tracking low-level memory calls through JavaScript APIs, since some scripts

could be encrypted or encoded. The former process is sequenced within a recurrent neural network (RNR) connection with long short-term memory (LSTM) units. As a result, a global precision of 93.04% was obtained. Likewise, in [12] Crypto-Aegis was built, which couples two aspects in a multi-adversary scope: insiders and outsiders. The first flank operates as a LAN network examiner, i.e., on the client side it can monitor the entire blockchain transaction pipeline, across the host's local network; the second one observes the outgoing load flows from the client to the remote PoW command and control center. In conjunction, adversarial observations are characterized as host-based and network-based traffic, conforming a dataset that is trained in its last stage by an ensemble algorithm (EA). It is shown that by profiling various network features of a web-based cryptojacking attack, in alignment with the distributive power of EA learning, it is possible to achieve an average F1 score of 96.00%.

Otherwise, in the work carried out in [48] it is assumed that by characterizing different CPU metrics: kernel-level usage, the start of processes and threads, interrupt and end time, and disk operations it is possible to fine-tune the detection of sites that send large amounts of batch processing from the browser to the operating system. Significantly, a portfolio of SL algorithms was used to evaluate the characterized CPU samples, on the basis of the following two-level classification algorithms: multiple-instance support vector machine (MISVM), random-subspace (RS) + decision trees (DT), and sequential minimal optimization (SMO) + SVM. On an overall average, a favorable result was obtained for the RS + DT algorithm by obtaining results in excess of 99.2% of accuracy. In view of WASM as the main indicator of cryptojacking traces, the authors in [16] developed a methodology known as MINOS, which dumps the sections of benign and malicious executable binaries, represented as gray-scale images. Then, a convolutional neural network (CNN) can learn from anomalous patterns, sensing them as malicious or benign. It is argued then, that it is possible to implement the methodology as a blocking extension in browsers such as Chrome, as the process is computationally lightweight and supported with an overall performance metrics above 98.97%.

To conclude with the related works that occupy ML as the main focus of their contribution, in [49], a method for characterizing Botnets with some cryptojacking patterns in physical IoT devices was developed by combining an adaptive filter with an AE and a scatter graph to identify the P2P nodes that congregate the commands from a control and command center, that pushes the requests as a distributed denial of service (DDoS) attack. It is shown that by identifying the root of a botmaster using network features and client resources, logistic regression (RL) and support vector machine (SVM) algorithms can achieve accuracy rates of up to 99.69%. Aggregating the above information, Table 1 describes most of the relevant work that has been developed to mitigate web-based cryptojacking. It is organized by including the type of scope, its algorithm/technique, the analyzed threat, and the type of defense employed.

Of the widely identified works tabulated in Table 1, ref. [8,29,37,39,40,42] contain some type of static or dynamic defense, particularly on the client side (host-based) [8,29,40,42], which is a great advantage for a first kick-off and triaging of raw cryptojacking observations, but in any case, it is possible to fail in the process of rapid assimilation of numerous samples that may interact with higher incidence, because its strength lies entirely in manually generated efforts to unveil the results. The two remaining hybrid works [37,39] employ SA and DA jointly, but continue to follow the so-called mechanism of comparison based on bit distances or similarity, leaving aside important aspects that can be better implemented through anomaly detection processes. In contrast, there are eight recommendations that make use of some type of intelligent mechanism or are ML-driven, of which four are host-based solutions [16,43,44,48], two network-based [12,38], and three hybrids [31,47,49].

Table 1. Related work on detection of cryptojacking and cryptomining threats in the web browser.

Authors	ML-Based	Algorithm/Technique	Type of Threat	Type of Defense
Aziz et al. [37]	No	Static behavioral signatures, embedded in perimeter security devices	Network traffic traces or binaries detected with cryptomining malware	Hybrid
Razali et al. [39]	No	Blacklists, browser behavioral analysis and dynamic content	Web-based cryptojacking	Hybrid
Eskandari et al. [29]	No	Dynamic analysis: blocking of excessive use of resources. Policies: consent of use	Web-based cryptojacking	Host-based
Rüth et al. [8]	No	WASM fingerprinting	Web-based cryptojacking	Host-based
Hong et al. [40]	No	Hybrid analysis: on the static side: SSBP; on the dynamic side: web content analysis	Web-based cryptojacking	Host-based
Burgess et al. [42]	No	Web crawling + customized regular expressions	Web-based cryptojacking	Host-based
Saad et al. [31]	Yes	Content-currency and code-based analysis + Fuzzy C-Means	Web-based cryptojacking	Hybrid
Berecz et al. [38]	Yes	SVM, Multi-Layer Perceptron(MLP) and RF	Anomalous sites and URLs, related to cryptojacking & Malicious email attachments	Network-based
Carlin et al. [43]	Yes	Random Forest (RF)	Web-based cryptojacking	Host-based
Wang et al. [44]	Yes	SVM	Web-based cryptojacking	Host-based
Liu et al. [47]	Yes	LSTM	Web-based cryptojacking	Hybrid
Caprolu et al. [12]	Yes	RF	Malicious cryptojacking patterns on network traffic	Network-based
Gomes et al. [48]	Yes	MISVM, RS + DT (Decision Trees), and SMO + SVM	Web-based cryptojacking	Host-based
Naseem et al. [16]	Yes	CNN	Web-based cryptojacking	Host-based
Kumar et al. [49]	Yes	SVM, RL, and CNN (Convolutional Neutal Network)	Cryptojacking as part of a botnet	Hybrid

The great advantage of AI or ML applications is an improvement for quickly portraying emergent cryptojacking scenarios, in addition to tools that can be integrated in an enhanced manner; however, there is still a question mark derived from the type of characteristics to be analyzed, firstly, some authors agree in giving more steadiness to the client side, since the computation of blocks resides in the operating system of the infected device; but, on the other hand, it is stated that there are important details that emerge in the communications network, since it is the entry point of the threat. Of the hybrid solutions using ML, the former [31] delegates more importance to client-side activity, concentrating its logs on outgoing network packets and subsequently on the effects of malware responses on the CPU, specifically on prolonged power consumption. In turn, the second one [47], is more oriented to handle several elements belonging to the content of the infected sites, which could be a great advantage when trying to categorize the type of cryptojacking families, as it is often performed for phishing menaces in accordance with natural language processing (NLP) methods. Conversely, this would not help in the construction of the dataset, by considering web content in co-occurrence with network and host features, since as a consequence, the dimension of the data could increase, magnifying the risk of misgeneralization and leading to a possibly slower analysis. The third and last one [49] is the only manuscript that applies a similar AE technique, but the pre-processing steps first lie in the operation of a botnet, assuming that cryptojacking attacks are derived from it, and in the course of the tests, the authors refer to shallow algorithms as the machinery for

detection. From this perspective, authors [50] who evaluated a similar threat, but within IoT spoofed nodes, emphasize that the content of the threat, e.g., printable characters, payloads, or encoding, is important for the initial recognition tasks, but not sufficient to assume that a malicious object is identifiable through these factors, since for experienced malicious actors, these are only the vehicle of the attack, and the activity occurs mostly at the client level, at the network layer and the incidents that converge in both.

After presenting the list of advances that have been made historically in terms of cryptojacking detection on the web, it can be summarized that there is currently no consensus on the type of features to evaluate in order to generalize the picture. Similarly, there is no clear context to address which pre-processing steps may be the most ideal to employ to improve data quality, and finally it has been shown that although shallow algorithms can quickly generalize the sample set, by their nature, they tend to work only on well-structured data, losing learning power if the proportion of information contains variable multidata [51].

For this reason, a multi-layered viewpoint (host-based and network-based) is proposed in this project, with a new scope in the characterization of the cryptojacking samples, as a fundamental element in a proper detection. Thus, it can be said that the following hypothesis arises:

Research hypothesis: by addressing the two feature selection flanks on which the cryptojacking attack vector extends, the host and the network, and by representing their best latent information employing algorithms that maximizes them, such as an AE and a deep dense neural network (DNN), the detection radius can increase significantly.

5. Proposed Framework

The workflow outlining the main aspects of this project is described in Figure 2. First, by considering the websites proposed in [42,43,52] and a curated list of malicious cryptomining scripts previously identified in [42], a catalog of malicious URLs was generated and automatically traversed using web-scraping techniques; each site was fingerprinted in terms of highlighting the existence of cryptojacking scripts by computing statistical NLP techniques. Then, sites that are likely to be suspicious are examined in a sandbox environment, where the behavior is dynamically monitored with the assistance of an instrumented navigator, that allows assessing network traffic and operating system traces. The outputs are then stored on a local data warehouse, that is to say, tabulated and presented as a dataset with associated class labels. Next, pre-processing techniques are applied, based on feature extraction and selection algorithms, using AE techniques. Further on, a subset of training data was prepared to train the DDNN algorithm, and finally, with a subset of test samples, the performance of the resulting classification model was measured in terms of the following performance metrics: Precision, Recall, and F1-score.

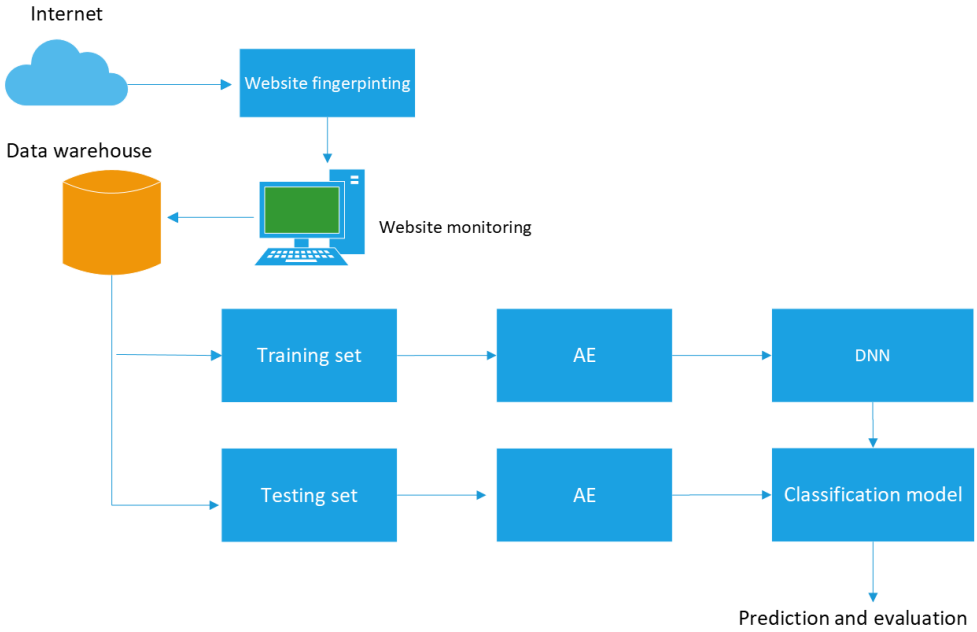


Figure 2. Workflow of the proposed methodology.

5.1. Data Collection

The data collection process starts with fingerprinting, which is a well-known initial detection process that correlates different data, versions, services, and some patterns when performing penetration testing (pentesting) tasks. In this case, cryptojacking, being a threat that resides mainly on websites, goes through an exploration process similar to phishing, with particular aspects that are listed below:

- Cybercriminals can create instances of illegal cryptocurrency mining within a few lines of source code in a script, which can slow down the fingerprint search based on simple text queries towards a broad set of websites. This also tends to happen when the script is part of another vulnerability exploit and the attacker attempts to encode or encrypt the lines in other schemes, to evade defenses.
- Often, malicious sites redirect requests within HTTP headers, negatively affecting the responses when capturing client–server bidirectional transfer of information. The answer is simple: if a traditional fingerprinting scope is set via web crawling, it may fail to properly follow the responses, as it may only return the first and not the forwarding paths. Some attackers time the loading of HTML or add dynamic content presentation mechanisms to avoid content query automation techniques. Other malicious actors add riddles, bogus captchas, or shorteners that only by constant human interaction will reveal the final destination.

For this reason, an analysis of data congregation between layers is proposed as shown in Figure 3. Each layer is described below:

- *Layer 1—Fingerprinting*: in this initial layer, sites that may contain signs of crypto-mining are captured. This stage is vital as it filters out those that will be useful for exploration and eventual characterization, discarding inappropriate storage of unwanted, duplicated, or error-coded sites.
- *Layer 2—Network-based data*: comprises the information flows of traffic transiting through the HTTP/s and TCP protocols. This allows the analysis of request–response patterns in site activity and tracking nested scripts on the site.
- *Layer 3—Host-based data*: is the monitoring of the website once it interacts with the browser. In this step, the activities of the web browser are implemented through an application that resembles human navigation, leaving aside the complications of a classic web crawler. Attention is paid to the actions that are triggered once the content is rendered and consequently the behaviors that are directed towards the CPU and

the processes that are activated for resource consumption. At this stage, suspicious samples that are not detected in the previous ones, due to their obfuscated or coded form, can be revealed.

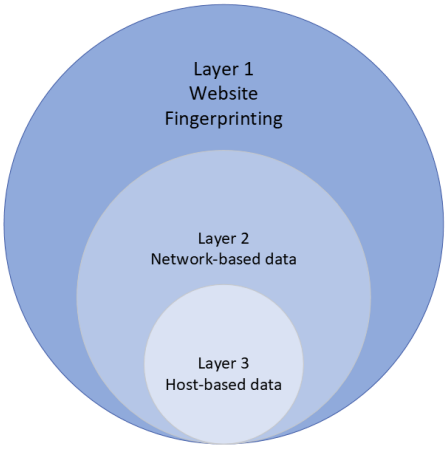


Figure 3. Proposed strategy for the capture of cryptojacking samples.

To run the instances for fingerprinting, analysis, network monitoring, and capture of host activity, a virtual environment was configured, with the following characteristics: two virtual machines (VMs) with Ubuntu 16.04 and Windows 10 operating systems, both with an Intel Core i7 processor and 16 GB of RAM. For HTTP crawling and TCP processing, native parsing libraries were used by means of Python 3.7 programming language. For its part, Selenium [53] web instrumentation tool served as the action automator of the fingerprinted sites, which inherently contains Chrome as the browser that resembles the human-oriented navigation activity.

In order to select the necessary sites to be crawled, dumps of databases with previously identified malware, were appended: firstly, the NoCoin adblock list [52], which is a series of 642 URLs with regular expressions, containing traces of common web-based cryptojacking families. It is actively handled to complement extensions such as AdBlock, uBlock Origin, and AdGuard; also conforming a native component of web browser policies such as Brave and Opera. Secondly, the compendium presented in [42] was integrated, which likewise, employs a list of malicious sites that have been previously sandboxed and tagged with some sign of illegal cryptomining. This breaks down into a list of 913,554 benign sites, 63,002 URLs that could not be accessed correctly or displayed some HTTP error code, 415 suspicious sites with some type of embedded malware, and 887 sites that were undeniably classified as malicious.

From a total of 2038 sites, a first fingerprinting traversal run was executed using Selenium, with the objective of extracting active cryptojacking script structures and therefore confirming if the site is still operating. For this, HTTP responses with uploaded .js object extensions were identified and captured. Afterwards, the asynchronous XMLHttpRequest loads were filtered out. Finally, a parsing process to HTML content, to select only DOM tags that matched a regular expression, was designed to highlight any element within the normative JavaScript syntax. Each discovered script was transformed at a character level using the TF-IDF (term frequency-inverse document frequency) algorithm, known to statistically encode words or letters based on their relevance and represented by weights (normalized values under the range of $[0, 1]$) comprised in a vector, of vocabulary length (number of encoded characters).

To filter the sites, each vectorized script was compared by cosine distance against the summary presented in [42], where 981 cryptojacking code fragments are presented. If the site does not present an error code or a timeout and the similarity is greater than 0.8, the presence of scripts is confirmed, labeled as cryptojacking and stored in the database, along with the DNS resolutions, domain names, IP addresses, and access nodes, with

which it interacts to complete the PoW. According to [54], the similarity values vary in the range $[0, 1]$, where 0 represents the absence of similarity and 1 the perfect comparison. The threshold is difficult to establish since 0.6 could be said to represent a 60% match. In this particular scenario the highest similarity was 0.91 and 0.13 the lowest, with an overall average of 0.82. Thus, 0.8 was left as the optimal criterion as a minimum similarity. For those that do not exceed the threshold, but could be suspicious, a second traversal recursive tracker was programmed to follow requests and links to the already identified lists from the first traversal run, if any of the following scenarios occur: unwanted downloads, pop-up flooding, or malvertising, URLs with encoded characters or pronounced length, constant redirects, and forced clicks. If a request matches, then the site is also labeled as cryptojacking and appended to the database. In a third traversal run, the previously confirmed and stored sites were used as the basis for customized queries on Public WWW [55] databases, which is a source code search engine that allows finding any alphanumeric fragment, signature or keyword over HTML, JS, and CSS documents across a wide selection of websites.

To complete the dataset with benign sites, a random selection was made from those listed in Alexa Top Domain List [41], since a large part of them share similar anatomies: active HTTPs configuration from common certificate authority companies, controlled redispatching to non-out-of-domain links or subdomains, presence of CSRF tokens in POST methods or X-CSRF-TOKEN header attributes and cross-domain policies. On the other hand, other security policies that they share are: JS execution is limited to interactions within the DOM, asynchronous uploads are restricted to the same host or domains with same-protocol policies and through verified whitelisting of social networks, payment APIs, animation scripts, and web development frameworks. The final dataset comprises 8000 random benign sites, and 8156 tagged and validated as cryptojacking. Table 2 tabulates the families of cryptojacking scripts, the top-level domains (TDLs), DNS, or IP addresses found, and the number of infected sites involved.

Table 2. Queries of malicious script patterns, top-level domains, and number of detected sites

Malicious Script	TLD/DNS/IP Targeted	Number of Infected Sites
Crypto-loot	crypto-loot. com, cryptoloot. pro, 104. 21. 56. 113 172. 67. 184. 223	355
CoinIMP	hostingcloud. *, free-content. *	1931
Minr	cnt. statistic. date. *, cdn. static-cnt. bid. *, ad. g-content. bid. *, cdn. jquery-uim. download. * minr. pw, cnt. statistic. date. *, ad. g-content. bid. *	692
DeepMiner/Coinhive	37. 187. 165. 17, 217. 182. 164. 9 217. 182. 164. 10	1979
Jsecoin	server. jsecoin. com, load. jsecoin. com, platform. jsecoin. com	2377
RubyMiner	internetresearch. is, dgnfd564sdf. com 203. 24. 188. 242	822

In the next step, to aggregate the features at the network and host layer, each infected and benign website was navigated for a period of 120 seconds, simulating a user's browsing; on Ubuntu (Linux) and Windows operating systems. Using the Processing Performance Tool (Windows) and HTOP (Ubuntu-Linux), processing signals from the web browser to memory, disk, and CPU were recorded every 30 seconds. Parallel to this, Wireshark protocol analyzer was focused on inspecting, filtering, and capturing network packets on TCP and HTTP protocols. Each analysis result was mapped with its respective site, class, and stored as a document in a separate collection in the data warehouse, which is the base of operations of the project. The final eighteen features are presented in Table 3, where their exploratory static values are described.

Table 3. Features comprising the data set.

Feature Name	Description	Type	Mean (μ)	std. (σ) †	min. ‡	25% §	50% §	75% §	max ‡
C1	Time in C1 is a subset of the total processor idle time	host-based	68.52	39.80	0	18.25	90.87	97.69	100.98
C2	Time at C2 is a state of lower energy and higher output latency than Time at C1	host-based	48.29	29.52	0	29.86	51.82	72.03	98.52
C3	Time at C3 is a lower energy state and higher output latency than Time at C2	host-based	9.27	7.39	0	1.80	8.85	15.77	22.83
I/O Data Operations	Speed at which the process is issuing read and write I/O operations	host-based	36.13	72.26	0	1.73	12.08	35.68	1795.31
I/O Data Bytes	Speed at which the process is reading and writing bytes in I/O operations	host-based	1.11×10^5	4.79×10^5	0	8.32×10^2	5.07×10^3	3.97×10^4	4.67×10^6
Number of subprocesses	Number of sub-processes that are currently active in a parent process	host-based	29.82	5.67	1	27	28	30	51
Time on processor	The total time, in seconds, that a process has been running	host-based	0.48	1.55	0	0	0.03	0.45	25.24
Disk Reading/sec	Speed of disk reading operations	host-based	5.33	18.07	0.04	0.75	1.71	5.90	1064.81
Disc Writing/sec	Speed of writing operations to disk	host-based	0.98	13.8	0	0	0	0.02	831.50
Confirmed byte radius	The ratio of Memory/Bytes committed and Memory/Confirmation limit	host-based	28.45	3.96	18.05	25.26	28.57	30.99	46.78
Percentage of processor usage	Elapsed time on the processor when an active thread is running	host-based	31.48	39.79	1.092	2.30	9.06	81.75	100.01
Pages Read/sec	Speed rate at which the disk was read in order to resolve hard page errors	hots-based	0.99	12.07	0	0	0.02	0.08	475.03
Pages Input/sec	Speed at which pages are written to disk to free up space in physical memory	host-based	0.01	0.08	0	0	0	0	3.91

Table 3. *Cont.*

Feature Name	Description	Type	Mean (μ)	std. (σ) †	min. ‡	25% §	50% §	75% §	max ‡
Page Errors/sec	This is the average number of pages with faults per second	host-based	2760.23	4905.27	16.14	353.27	685.07	1371.68	98,031.40
Bytes Sent/sent	The rate at which bytes leave the browser's HTTP requests	network-based	538.14	1766.38	1.19	33.39	66.87	218.10	98,097.78
Received Bytes (HTTP)	Speed of bytes arriving to the browser's HTTP responses	network-based	15,394.97	80,792.11	1.55	27.29	86.24	547.75	4,812,144.22
Network packets sent	Speed of sending packets in the TCP protocol	network-based	4.54	18.60	0.02	0.39	0.71	1.53	1118.87
Network packets received	Packet reception speed over the TCP protocol	network-based	11.84	54.94	0.02	0.33	0.62	1.71	3183.70

† displays the value of the standard deviation; ‡ represent the maximum and minimum values; § shows the percentile values at 25%, 50% (median), and 75% of the data, sorted in ascending order.

5.2. Data Pre-Processing

This stage describes the pre-processing of the data, where data cleaning, transformation, and normalization techniques are applied. The process starts with exploratory and cleaning analysis, aiming to impute missing, null, or unprocessable data via last observation carried forward (LOCF) [56], which performs a statistical longitudinal compensation that replaces the errors with the last carried forward observation. The dataset is then subjected to a Z-score normalization process [57], due to the existence of outliers. This is a common phenomenon when converging measurement data from different sources, generating a trend prone to bias-variance effects, such as demonstrated in Table 3. Z-score normalization is denoted in Equation (1).

$$X_{zscore} = \frac{X - \mu}{\sigma}, \tag{1}$$

where X is the raw dataset, μ is the mean, σ is the standard deviation, and X_{zscore} is the new normalized data.

Once the data are normalized, a novel compression and dimensionality reduction technique, namely, the autoencoder [25] is proposed. This is an unsupervised learning algorithm, which uses a backpropagation and compression technique to reconstruct the output as closely as possible to the input of the network, in other words, with a minimum reconstruction error that can be used for feature extraction, with a latent representation of the original characteristics, while minimizing the effects of linearity. AE applications have been successfully applied in cybersecurity areas, as they have in common, the handling of raw and noisy information of web attacks, forensic chains, indicators of compromise, malicious techniques and tactics, and substantially any security disruption event that must be quickly transformed and evaluated. As an example of their application, in [58], two scenarios were selected: network-based anomaly intrusion detection and malware classification. The tests showed rapid progress in detection ratios, without the need for large pre-processing steps. In the same direction, several authors have applied different AE assemblies to improve detection techniques based on anomalies occurring in IoT devices [59], security situation assessment [60], and for cyber–physical system (CPS) environments [61]. Having mentioned the applications of an AE, Figure 4 displays its formal structure.

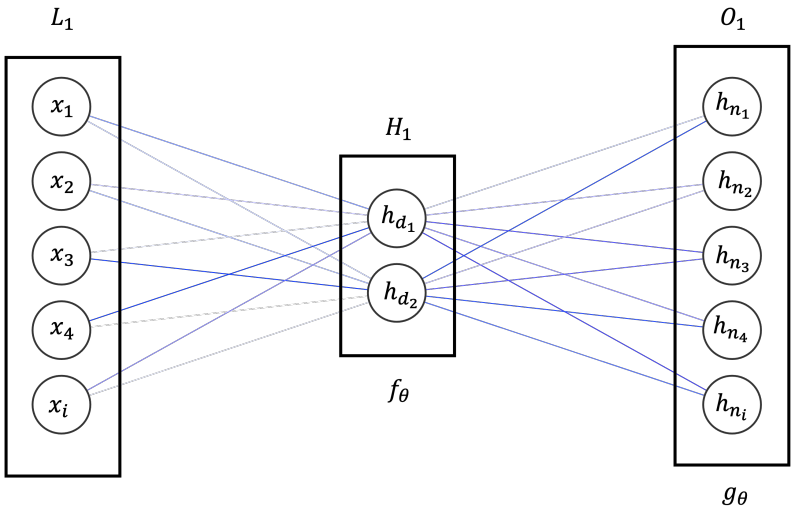


Figure 4. Graphical representation of an AE.

When the AE reconstructs the input, it learns from an identity function that maps the input against the output. Thus, it comprises an input layer (L_1), a hidden layer (H_1) and an output layer (O_1), where the hidden units at H_1 layer learn from a dimensionally reduced representation of the input, usually with fewer neurons or units than L_1 [62]. In

the encoding phase [63], the compact hidden representation, denoted as vector h_d, of the input vector x with d characteristics is given by Equation (2).

$$h_d = f_{\theta(x_d)} = \gamma_1(W_{x_d} + b_1), \tag{2}$$

where $f_{\theta(x_d)}$ is the mapping function, W_{x_d} is the $p \times p$ weight matrix with p hidden units, b_1 is the bias vector, θ is the parameterized set, and γ_1 is the encoder activation function. In the decoding phase, the reconstructed vector of dimension d is calculated by mapping back the compressed hidden representation or vector h_n, as shown in Equation (3).

$$h_n = g_{\theta(h_d)} = \gamma_2(W_{h_d} + b_2), \tag{3}$$

where $g_{\theta(h_d)}$ is the inverse mapping function, γ_2 and b_2 the decoder activation function and bias, respectively. The optimized mean squared error function is one of the most commonly used loss functions when it comes to AEs to measure the reconstruction error L. Given that $f_\theta : x \to h_d$ constructs the input and $g_\theta : h_d \to x$ decodes the output, then the error can be minimized as denoted in Equation (4).

$$\arg\min_{f_\theta, g_\theta} ||x - (f_\theta \circ g_\theta)||^2 \tag{4}$$

Equation (4) can be rewritten as follows:

$$L(W, b, x, h_n) = ||h_{W,b}(x^{(i)}) - h_n^{(i)}||^2, \tag{5}$$

accordingly, in Equation (5), $x^{(i)}, h_n^{(i)}$ are the coded and decoded samples of the training set; $h_{W,b}$ is the hypothesis parameterized to a weight matrix $W^{p \times p}$ and the bias b.

One of the important points to consider about the architecture of an AE is that the coding/decoding capacity can be very high, this is usually caused by a compromise when h_d is greater than x, leading to a serious problem of learning the final representations. This is solved by modeling the distribution complexity of the model, using regularization by the ℓ_1 norm. Thus, the AE will be able to set the level of noise or missing values, even if the capacity is not large enough to learn a trivial identity function. One of the most widely used AE configurations for classification problems, with regularization elements, is the sparse auto encoder (SAE), where only some nodes are forced to activate when an input feeds the network. With this, features will still be optimally represented by reading latent representations, reducing redundant information. Figure 5 shows the graphical form of an SAE.

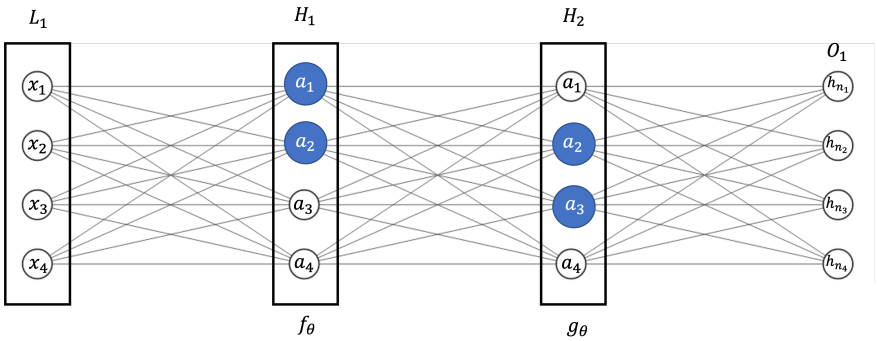


Figure 5. SAE architecture. The SAE-type AE network can work with input and output copies in conjunction with supervised or unsupervised targets. Here the highlighted $a_1, \ldots, a_k$ neurons represent those activated in the hidden layers.

For the purpose of noise reduction and redundancy of the normalized samples, the SAE regulated model is used in this work, adding the penalty term $\ell_1(h_d, x)$ in the loss function, as shown in Equation (6).

$$obj = L(W, b, x, h_n) + \ell_1(h_d, x), \tag{6}$$

where obj corresponds to the objective of the loss function with the shrinking term of the ℓ_1 norm. Table 4 shows the values used to configure the SAE. According to [64], the hidden layer H_1 can occupy $\frac{2}{3}$ units proportional to the length of the neurons in the input layer L_1; similarly, the resized output layer O_1 contains a proportion of units at least half the length of the input layer L_1.

Table 4. Settings and parameters for SAE construction.

General Configuration Properties	Layer	Units	Activation Function
Optimizer = ADAM			
Loss function: MSE (Equations (5) and (6))	L_1	18	ReLU
Penalty value for $\ell_1 = 10 \times 10^{-5}$	H_1	14	ReLU
Epochs = 100	O_1	9	Softmax
Batch size = 32			
Validation split = 20%			

In order to improve the inter-neuronal learning rate of the SAE, the hyper-parameter ADAM (adaptive moment estimation) was added [65]; an optimization algorithm that works in conjunction with the gradient method, becoming quite efficient given the existence of large amounts of parameters in the data. It combines and extends the properties of AdaGrad (adaptive gradient algorithm) and RMSProp (root mean square propagation), which allow increasing the quality of the learning radius in the presence of sparse gradients, by averaging the speed of change (the average of recent gradients). The great advantage is that it requires little memory, making the adaptation ideal for data sets that are measured in a short time.

5.3. Deep Dense Neural Networks

Dense neural networks are an architecture of deep, biologically inspired models, consisting of one input L_1, one output O_1 and one or more hidden layers $H_1, \ldots H_n$, with the ability to process information of a non-linear nature, to solve supervised or unsupervised learning problems [66]. As the name indicates, the neurons are densely connected, with each neuron receiving information from the previous layer, similar to an AE, providing an initial abstraction layer that is the basis for networks such as CNNs and RNNs. Their great ability to generalize a scenario has made them preferred over shallow algorithms [22,38], which need more processing tasks before learning from the samples that encompass the context. Figure 6 depicts the architectural form of a DDNN.

For stability purposes, a DDNN with rectified linear units (ReLU) activation functions with three hidden layers is proposed to establish a non-linearity criteria following a regularized dropout layer and a sigmoid output layer, since the expected values will be binary $y \in [0, 1]$, where 0 represents a benign site and 1 cryptojacking. The final architecture of the DDNN, used for this project, is described in Figure 7 and Table 5.

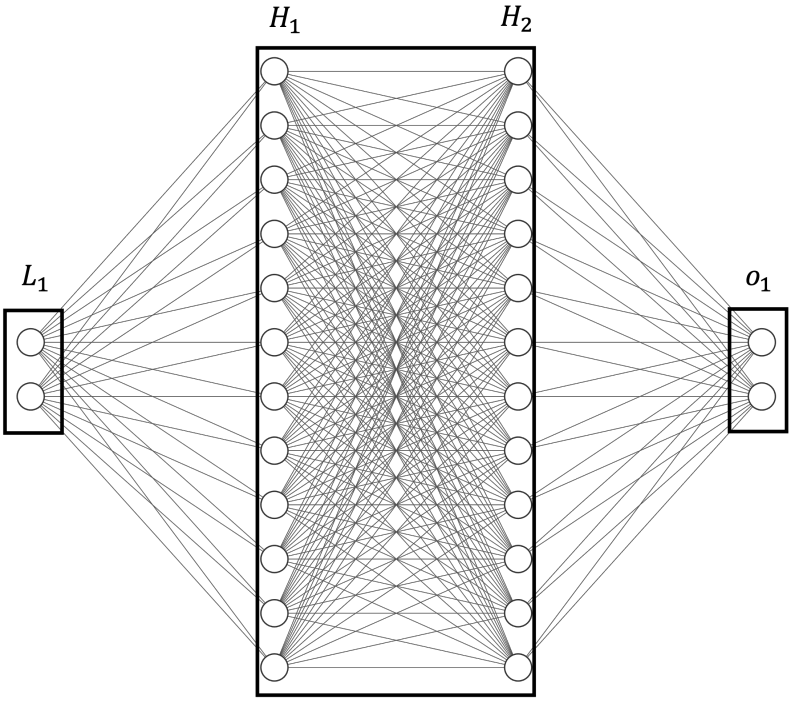


Figure 6. Conceptual diagram of a feed-forward, fully connected DDNN. DDNNs have been applied in multiple cyber-security tasks [12,26,43,44,48,49,67] because of their ability to effectively analyze different backgrounds, types of values, and measurements in the collected data, often from sources that are presented in real time. The propagation of desired feature information between multiple neurons in the hidden layers H_1 and H_2 is strengthened, maximizing the recognition capability of a sample, in this case, cryptojacking.

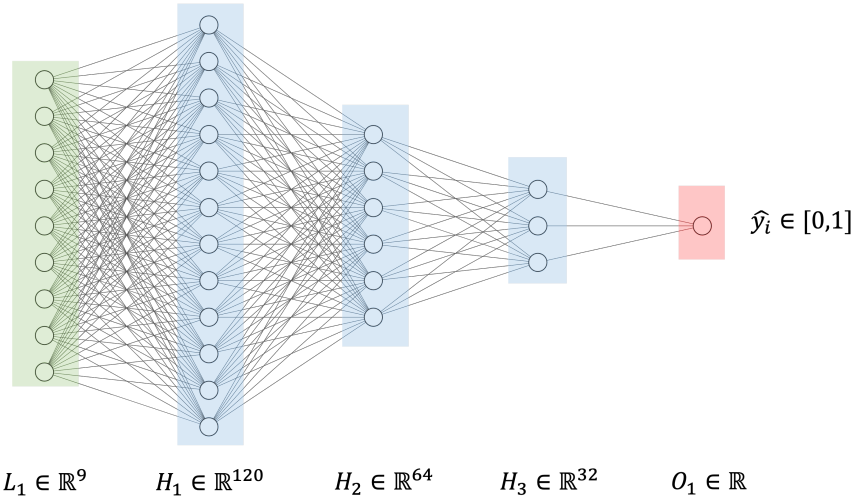


Figure 7. Final DDNN architecture, which consists of an input layer $L_1 \in \mathbb{R}^9$, three hidden layers $H_1 \in \mathbb{R}^{120}$, $H_2 \in \mathbb{R}^{64}$, $H_3 \in \mathbb{R}^{32}$ and a final output layer $O \in \mathbb{R}$.

Table 5. Settings and parameters for SAE construction.

General Configuration Properties	Layer	Units	Activation Function
	L_1	9	ReLU + weight constraint = 5
Optimizer = ADAM	H_1	120	ReLU
Loss function: Binary Cross Entropy	H_2	64	ReLU
Epochs = 100	H_3	32	ReLU
Batch size = 32	Batch normalization layer		
Validation split = 20%	O_1	1	Sigmoid

In Table 5, three hyper-parameters are introduced. The first concerns weight constraints, which are a way of adjusting weight sizes against a pre-established threshold, ideally functioning as regularization, ensuring that weights remain small during training, avoiding fading, or decay problems. The second with the training time, which through the normalization layer, helps to increase the learning coefficient, making it faster to learn and adjust weights, while ensuring that the input to the layer maintains a good distribution within the mean and standard deviation of the units. Thirdly, a loss function, called binary cross-entropy, is applied, which arises from taking advantage of the probabilistic relationship of the desired class distributions predicted by the DDNN. The LOSS function is denoted in Equation (7).

$$LOSS = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \mathbf{log}(\hat{y}_i) + (1 - y_i) \cdot \mathbf{log}(1 - \hat{y}_i), \tag{7}$$

where y_i is the expected class, $\hat{y}_i$ the predicted output of the DDNN and N the size of the output.

6. Results and Discussion

This section describes the classification results of the SAE + DDNN, to demonstrate the viability of the algorithms presented in this project. For the experimental tests, it was decided to first train an unnormalised set X directly towards the DDNN algorithm, with raw samples and then by transforming each one, via SAE characterization. Furthermore, using the normalized set X_{zscore}, by feeding raw normalized samples towards the DDNN inputs, and subsequently, by applying the SAE encoder. Both scopes engage training sets $\{X_{train} \in X$ and $X_{train_z} \in X_{zscore}\}$ with 80% of random samples without replacement, from which validation subsets $\{X_{val} \in X_{train}$ and $X_{val_z} \in X_{train_z}\}$ with 20% samples to obtain preliminary metrics in terms of accuracy, during each epoch. In due course, the test subsets are divided $\{X_{test} \in X, X_{test_z} \in X_{zscore}\}$, with 20% of observations for the final DDNN model evaluation. The metrics used to display the performance values are Precision, Recall, and F1 score, as described in Table 6.

Table 6. Proposed performance metrics. It is important to note that the target class 1 is the site infected with cryptojacking, therefore TP represents the true positives and FP the false positives. Analogously, the non-target class 0 represents the benign sites, of which TN is identified as the true negatives and FN as the false negatives, respectively.

Evaluation Metric	Description	Mathematical Description
Precision	It measures the quality of the model. It is the fraction of relevant observations relative to the total number of classified instances of benign and cryptojacking sites.	**Precision** $= \frac{TP}{TP + TF}$
Recall	Describes the fraction of classified relevant samples out of the total number of relevant instances.	**Recall** $= \frac{TP}{TP + FN}$
F1-score	It is the harmonic between Precision and Recall, it measures whether the model performs as expected, by compensating the PF and FN rates.	**F-1** $= 2 \cdot \frac{\textbf{Precision} \cdot \textbf{Recall}}{\textbf{Precision} + \textbf{Recall}}$

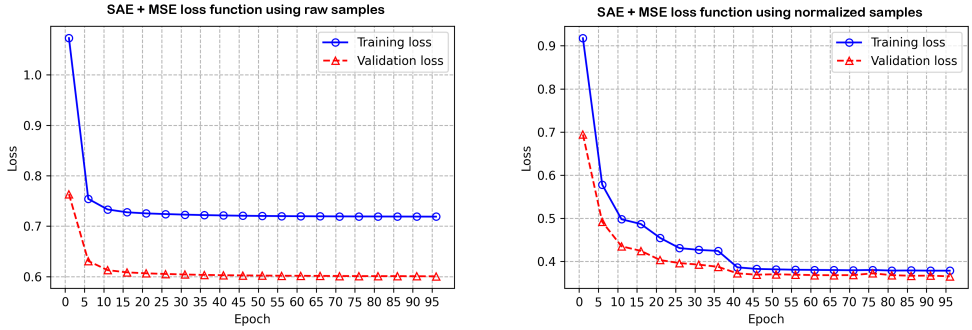
Table 7 shows the results obtained through the above-mentioned data set configurations.

Table 7 shows an improvement in pressure for raw X_{test} and normalized samples X_{test_z} with the SAE + DDNN configuration, with an average score of 99.41% and 90.03%, respectively; however, for raw samples X_{test}, a lower Recall rate, with a value of 24.59% was presented. This is an effect explained in [64], as the reconstruction of the output is facilitated with normalization processes, as it is well known that high variability ensembles are prone to fail when copying the information in the bottleneck layer, that is called as, reconstruction error effect. This problem can be commonly observed by plotting the MSE cost function of the SAE in terms of training data losses versus model validation losses. Two events

occur during this representation, when the training set is not sufficiently representative or the validation set presents problems to generalize each sample–characteristic relationship. Both conditions generate a gap in the convergence of the reconstruction of the SAE inputs, causing a decay in the output layer, finally leading to underfitting, as portrayed in Figure 8a. In turn, Figure 8b shows the stability of the losses during training and validation in regard to X_{val_z} and X_{train} from epoch 80 onwards, where the two are coupled in a constant decrease.

Table 7. Performance results obtained after evaluating the DDNN + SAE model.

Testing Set	Configuration	Target Class	Precision	Recall	F1-Score
X_{test}	DDNN	cryptojacking	83.61%	51.44 %	63.69 %
X_{test}	DDNN	benign site	27.87%	17.15 %	21.23 %
Average	DDNN		55.74%	34.29%	42.46%
X_{test}	DDNN +SAE	cryptojacking	99.57%	12.57 %	22.33 %
X_{test}	DDNN + SAE	benign site	80.49%	36.61 %	28.97 %
Average	DDNN + SAE		90.03%	24.59%	25.65%
X_{test_z}	DDNN	cryptojacking	81.40%	99.29 %	89.46 %
X_{test_z}	DDNN	benign site	99.70%	91.38%	95.36 %
Average	DDNN		90.55%	95.33%	92.41%
X_{test_z}	DDNN + SAE	cryptojacking	99.41%	99.78%	99.59 %
X_{test_z}	DDNN + SAE	benign site	99.42%	98.43%	98.92%
Average	DDNN + SAE		99.41%	99.10%	99.25%



(**a**) (**b**)

Figure 8. Comparison of loss function curves using MSE, in contrast to raw and normalized samples, subject to SAE reconstruction.

At this point, the proposed SAE+DDNN architecture is initially compared with shallow state-of-the-art algorithms, as described in Section 4. To run the comparative tests, a grid-search cross-validation [45] approach is considered, which is a method of hyper-parameter tuning that seeks to maximize the performance of supervised algorithms by setting a series of values prior to training. The grid combines a cross-validation with k iterations, to calculate the average ability of the algorithms under test, towards a persistent accuracy. To find the optimal subset, it was formulated the approach described in Equation (8).

$$\lambda^* = \arg\min_{\lambda} \{F(L(A, X_{zscore}, \lambda))\}, \tag{8}$$

thus then, L denotes the loss function, F is the objective function, A is the algorithm to be submitted, X_{zscore} is the normalized training set, where each sample $x_i \in X_{zscore}$ is a fixed vector $\mathbb{R}^{1\times 18}$, and λ the hyper-parameters to be evaluated, resulting in a subset λ^* of optimal parameters. Table 8 lists the authors, algorithms, and hyper-parameters to be compared. Likewise, from the works that present applications of DL algorithms, a base architecture was arranged, to resemble as close as possible the authors scopes. The configurations used for training and comparison are tabulated in Table 9. It should be

mentioned that the classification strategy for the LSTM network employs a many-to-one design, since this work does not consider a sequential network, in this sense, for the CNN a fixed vector was used as with the shallow algorithms, since the samples can hardly be coupled to the dimensions of an image with 2D spatial layers for the convolution of graphs.

Table 8. List of selected authors who have presented solution proposals regarding cryptojacking, using shallow algorithms. The λ hyper-parameters to be evaluated and then optimized are described.

Authors	Algorithm	Hyper-Parameters (λ)	Values
Saad et al. [31]	Fuzzy C-Means	fuzzifier (m): Controls how fuzzy the inter-group boundaries will be	$\{0.5, 5\}$
Berecz et al. [38] and Wang et al. [44]	SVM	Penalty parameter C of error term	$\{10, 100\}$
	SVM	Type of division (div)	One-vs-one, One-vs-rest
	SVM	Kernel type (ker)	Polynomial, Linear, RBF
Berecz et al. [38]	MLP	Hidden layer sizes ($H_1 \in \mathbb{R}^n$)	$32, 64, 128$
	MLP	Activation Functions (act)	Sigmoid and hyperbolic tangent
	MLP	Weight optimization solver (sol)	ADAM and Stochastic Gradient Descent
Berecz et al. [38], Carlin et al. [43] and Caprolu et al. [12]	RF	Attribute selection method (asm)	GINI Index and Entropy
	RF	Max tree depth (mtd)	$\{1, 15\}$
	RF	Number of features to consider for best split ($nfBS$)	$\{2, 18\}$
	RF	Minimum number of samples required to be at leaf node ($mnsLN$)	$\{1, 30\}$
	RF	Minimum number of samples required to split internal nodes ($mnsIN$)	$\{1, 30\}$
	RF	Minimum number of trees in forest ($mnTF$)	$\{1, 5\}$
Gomes et al. [48]	MISVM	Penalty parameter C of error term	$\{1, 10\}$
	MISVM	Kernel type (ker)	Polynomial, Linear, RBF
	MISVM	Number of bags (nb)	$10, 20, 30, 40, 50$, with 100 samples per class each
	RS + DT	Number of estimators (trees)	$2, 10$
	RS + DT	Maximum number of features in each estimator (mnFE)	$2, 9$
	SMO + SVM	Penalty parameter C of an error term	$\{100, 1000\}$
	SMO + SVM	Kernel type (ker)	Polynomial, Linear, RBF
Kumar et al. [49]	LR	Inverse of regularization strength of term C	$\{0.1, 1\}$
	LR	Norm selected to regularize the cost function (ℓ_n)	ℓ_1 and ℓ_2
	LR	Optimization algorithm (opt)	Linear, LBFGS, SAG, SAGA

Table 9. List of authors who presented cryptojacking solution proposals, using DL algorithms. The layers and units used for the construction of the models are described.

Authors	Algorithm	Configuration
Naseem et al. [16] and Kumar et al. [49]	CNN	1. 1D convolution layer $L_1 \in \mathbb{R}^{1\times 18}$ with 64 filters, a kernel size with 3 units and as an activation function ReLU 2. First pooling layer with two units 3. 1D convolution layer H_1 with 32 filter, a kernel size with 3 units and as an activation function ReLU 4. Second pooling layer with two units 5. A dense hidden layer $H_2 \in \mathbb{R}^{32}$ with a ReLu activation function 6. The output layer $O_1 \in \mathbb{R}$ with a Sigmoid activation function.
Liu et al. [47]	LSTM	1. LSTM input layer 1. $L_1 \in \mathbb{R}^{1\times 18}$ with 32 units 2. One Dropout layer with a radius of .2 3. LSTM hidden layer $H_1 \in \mathbb{R}^{1\times 18}$ with 16 units 4. The output layer $O_1 \in \mathbb{R}$ with a Sigmoid activation function.

From Table 9, a general configuration was proposed comprising: the loss function via binary cross-entropy, ADAM optimization, and Accuracy as validation metrics. The overall results, with the aforementioned performance metrics, can be observed in Table 10.

Table 10. Comparison of state-of-the-art results after applying the grid search method. A significant performance improvement can be noted in the proposed SAE + DDNN technique.

Algorithms	λ^*	Precision	Recall	F1-Score	Training Time (h:m:s)
Fuzzy C-Means [31]	$m = 1.5$	95.71%	29.30%	42.35%	00:02:15
SVM [38,44]	$C = 10$, div = One-vs-rest and ker = RBF	72.47%	100.00%	84.04%	0:17:11
MLP [38]	$H_1 \in \mathbb{R}^{128}$, act = Sigmoid and sol = ADAM	92.90%	92.62%	92.32%	00:01:25
RF [12,38,43]	asm = GINI, mtd = 5, $nfBS = 9$, $mnsLN$ = 17, $mnsIN = 8$ and $mnTF$ = 4	94.67%	93.56%	93.74%	00:03:21
MISVM [48]	$C = 10$, ker = RBF and nb = 100	96.38%	76.03%	85.01%	00:29:37
RS + DT [48]	$trees = 4$ and $mnFE$ = 4	94.09%	87.87%	90.38%	00:02:27
SMO + SVM [48]	C = 1 and ker = RBF	69.79%	61.39%	65.11%	00:31:43
RL [49]	$C = 0.9$, opt = LBFGS and $\ell_n = \ell_2$	99.13%	98.00%	98.57%	00:01:45
CNN [16,49]	n/a	86.41%	83.23%	80.48%	00:07:55
LSTM [47]	n/a	81.22%	51.34%	50.65%	00:04:22
This work (SAE + DDNN)	n/a	99.41%	99.11%	99.26%	00:01:27

As shown in Table 10, after brute-forcing different hyper-parameters around the grid search technique, the algorithms that are closer to neural behavior, MLP [38], and RL [49], perform better on the F1 score ratio and reduced the training time. Although the LSTM network [47] also falls within this concept, its recurrent engineering could be better exploited by considering a scenario more focused entirely on content-based detection, but, as described in Section 5, an NLP-based fingerprinting process is more feasible, since it saves time in the initial recognition, without increasing the number of features in the dataset. The CNN [16,49] architecture achieves considerable results, but the transformation of samples to images and then to fixed vectors may degrade the ability to be spatially invariant to one-dimensional data such as those collected in this project. The maximum margin algorithms, SVM [38,44], and SMO + SVM [48], behaved with average performance, a major disadvantage is that the training stage requires considerable time and the search for suitable values for the penalty values C, increases the desired complexity. Of these, it is noted that the MISVM algorithm [48] adds more processing time, since the samples must be grouped and fixed in length in labeled bags, if the dataset is not large enough, the algorithm will end up misinterpreting the classification judgment for each bag, leading to major underfitting problems. The ensemble algorithms, RF [12,38,43], and RS + DT [48], on the other hand, obtained a timely performance in terms of F1 score rates, in addition to a fast training utilization, which is undoubtedly a clear example of why the authors decided to select them as their base classification engine, since both algorithms share in common the selection of random features subsets, resulting in multiple learners of low correlation, even in high dimensional ensembles. The fuzzy C-means [16] algorithm has an effective learning time, but its complexity increases, as with the maximum margin algorithms, when the fuzzifier m obtains small number ranges, leading to a large number of iterations for seeking the a priori number of specified clusters: benign sites and cryptojacking. It is demonstrated then, that with the development of SAE + DDNN it is possible to improve the performance of several works mentioned above and with a training speed similar to MLP. This is expected, since the algorithm can also be considered as deep, due to the number of neurons implemented in the hidden layer, and it could even be said that DDNN is an extension of MLP.

7. Conclusions

Cryptojacking is a stealthy threat that constantly affects users, who without realizing it, fall under the traps of malicious actors, who wait to consume the victim's resources without warning, in order to mine illicit operations towards a PoW. This paper proposes an ML-

based solution that employs a holistic view of cryptojacking, concentrating its effort on two major attack surfaces: the network, which is the entry point of the threat, and the host, which is where the malware payload is executed and disseminated. Considering the sophistication of several new strains, it is possible to characterize cryptojacking appropriately utilizing host and network features. To achieve this goal, it was first proposed a traversal web crawl towards possibly infected websites to perform the well-known fingerprinting process, using statistical comparison techniques. This reduces the dimensional size of the dataset and helps to follow a line of recognition similar to pentesting tests. This made it possible to analyze important sites and rule out several false positives or URLs with inaccessibility codes. Each site was subjected to a controlled environment that demonstrated that it is possible to capture evidence of cryptojacking by presenting activity on the network and within the victim's operating system once the attack is triggered. To represent the most important information, a novel feature extraction and selection technique called SAE was employed, which reduces dimensionality while retaining the best latent data for variant inputs such as those in the proposed dataset. The compressed and normalized outputs fed a DDNN, which, thanks to its deep design, allowed higher performance rates, compared to other state-of-the-art works, with a shorter training time. The values included are, for Precision 99.41%, for Recall 99.10%, and for F1 score 99.25%, with an approximate training time of one minute and twenty seconds. With the detailed metrics, one could recommend the development of detection tools at the browser level or as an addition to antivirus systems or industrial devices such as IPS and IDS, however, it must also be considered that there are limitations to carry them out, which are addressed in two key points of improvement. Firstly, it will be necessary to compile the model in such a way that a baseline detection with a sensing agent can recognize and classify the samples, to produce an alert or trigger a mitigation/remediation policy over minimal or restricted environments. The second, is to consider the detection of cryptojacking binaries and executables; here the procedure would be similar to the one developed in the present project, but at a lower level of abstraction. It should be considered the addition of a bilateral LSTM communication network, where the operating system API calls would extend the dataset and can be merged as a single row of network, host and operating system process activities, and calls.

Author Contributions: A.H.-S., G.S.-P., L.K.T.-M., J.O.-M., J.P.-P., J.-G.A. and L.J.G.V. contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101021801.

Acknowledgments: The authors thank the National Science and Technology Council of Mexico (CONACyT) and the Instituto Politécnico Nacional for the financial support for this research. This work has also received funding from UCM Projects THEIA (FEI-EU-19-04), THEIA I (FEI-EU-21-01) and THEIA II (FEI-22-01).

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

References

1. Brunnermeier, M.K.; James, H.; Landau, J.P. *The Digitalization of Money*; National Bureau of Economic Research: Cambridge, MA, USA, 2019; pp. 1–35.
2. Buraga, S.C.; Amariei, D.; Dospinescu, O. An OWL-Based Specification of Database Management Systems. *CMC-Comput. Mater. Contin.* **2022**, *70*, 5537–5550. [CrossRef]
3. Prabakaran, D.; Ramachandran, S. Multi-Factor Authentication for Secured Financial Transactions in Cloud Environment. *CMC-Comput. Mater. Contin.* **2022**, *70*, 1781–1798. [CrossRef]
4. Yuan, Y.; Wang, F.Y. Blockchain and cryptocurrencies: Model, techniques, and applications. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, *48*, 1421–1428. [CrossRef]
5. Dospinescu, O.; Caramangiu, M.E. The Key Success Factors for an M-Learning Cryptocurrency Application. *Inform. Econ.* **2018**, *22*. [CrossRef]
6. Sakas, D.P.; Giannakopoulos, N.T.; Reklitis, D.P.; Dasaklis, T.K. The Effects of Cryptocurrency Trading Websites on Airlines' Advertisement Campaigns. *J. Theor. Appl. Electron. Commer. Res.* **2021**, *16*, 3099–3119. [CrossRef]

7. Mestiri, H.; Barraj, I.; Mohamed, A.A.; Machhout, M. An Efficient AES 32-Bit Architecture Resistant to Fault Attacks. *CMC-Comput. Mater. Contin.* **2022**, *7*, 3667–3683. [CrossRef]
8. Rüth, J.; Zimmermann, T.; Wolsing, K.; Hohlfeld, O. Digging into browser-based crypto mining. In Proceedings of the Internet Measurement Conference, Boston, MA, USA, 31 October–2 November 2018; pp. 70–76.
9. Sanz-Bas, D.; del Rosal, C.; Náñez Alonso, S.L.; Echarte Fernández, M.Á. Cryptocurrencies and Fraudulent Transactions: Risks, Practices, and Legislation for Their Prevention in Europe and Spain. *Laws* **2021**, *10*, 57. [CrossRef]
10. Arık, Y.D.; Ertuğrul, M. Is the Energy-Hungry Bitcoin Beneficial for Portfolio Risk Reduction? In *Multidimensional Strategic Outlook on Global Competitive Energy Economics and Finance*; Emerald Group Publishing: Bradford, UK, 2022.
11. David, L.; Charles, M.; Michelle, M. *The Illicit Cryptocurrency Mining Threat*; Cyber Threat Alliance: Arlington, VA, USA, 2019; pp. 1–25.
12. Caprolu, M.; Raponi, S.; Oligeri, G.; Di Pietro, R. Cryptomining makes noise: A machine learning approach for cryptojacking detection. *arXiv* **2019**, arXiv:1910.09272.
13. Jayasinghe, K.; Poravi, G. A survey of attack instances of cryptojacking targeting cloud infrastructure. In Proceedings of the 2020 2nd Asia Pacific Information Technology Conference, Bali Island, Indonesia, 17–19 January 2020; pp. 100–107.
14. ENISA ETL2020—Cryptojacking. Available online: https://www.enisa.europa.eu/topics/threat-risk-management/threats-and-trends/etl-review-folder/etl-2020-cryptojacking (accessed on 19 December 2021).
15. ENISA Threat LANDSCAPE 2021. Available online: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2021/ (accessed on 19 December 2021).
16. Naseem, F.; Aris, A.; Babun, L.; Tekiner, E.; Uluagac, S. MINOS: A lightweight real-time cryptojacking detection system. In Proceedings of the 28th Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 21–25 February 2021.
17. Forestiero, A. Metaheuristic algorithm for anomaly detection in Internet of Things leveraging on a neural-driven multiagent system. *Knowl. Based Syst.* **2021**, *228*, 107241. [CrossRef]
18. Forestiero, A. Bio-inspired algorithm for outliers detection. *Multimed. Tools Appl.* **2017**, *76*, 25659–25677. [CrossRef]
19. Kim, H.; Park, J.; Kwon, H.; Jang, K.; Seo, H. Convolutional Neural Network-Based Cryptography Ransomware Detection for Low-End Embedded Processors. *Mathematics* **2021**, *9*, 705. [CrossRef]
20. Caprolu, M.; Raponi, S.; Oligeri, G.; Di Pietro, R. Cryptomining makes noise: Detecting cryptojacking via Machine Learning. *Comput. Commun.* **2019**, *171*, 126–139. [CrossRef]
21. Kharraz, A.; Ma, Z.; Murley, P.; Lever, C.; Mason, J.; Miller, A.; Bailey, M. Outguard: Detecting in-browser covert cryptocurrency mining in the wild. In Proceedings of the World Wide Web Conference (WWW'19), San Francisco, CA, USA, 13–17 May 2019; pp. 840–852.
22. Saad, M.; Khormali, A.; Mohaisen, A. Dine and dash: Static, dynamic, and economic analysis of in-browser cryptojacking. In Proceedings of the 2019 APWG Symposium on Electronic Crime Research (eCrime), Pittsburgh, PA, USA, 13–15 November 2019; pp. 1–12.
23. Darabian, H.; Homayounoot, S.; Dehghantanha, A.; Hashemi, S.; Karimipour, H.; Parizi, R.M.; Choo, K.K.R. Detecting cryptomining malware: A deep learning approach for static and dynamic analysis. *J. Grid Comput.* **2020**, *18*, 293–303. [CrossRef]
24. Zhao, J.; Jing, X.; Yan, Z.; Pedrycz, W. Network traffic classification for data fusion: A survey. *Inf. Fusion* **2021**, *72*, 22–47. [CrossRef]
25. Pinaya, W.H.L.; Vieira, S.; Garcia-Dias, R.; Mechelli, A. *Machine Learning. Methods and Applications to Brain Disorders*; Academic Press: Oxford, UK, 2020; pp. 193–208, ISBN 978-0-12-815739-8.
26. Hemalatha, J.; Roseline, S.A.; Geetha, S.; Kadry, S.; Damaševičius, R. An Efficient DenseNet-Based Deep Learning Model for Malware Detection. *Entropy* **2021**, *23*, 344. [CrossRef]
27. Cho, H. ASIC-resistance of multi-hash proof-of-work mechanisms for blockchain consensus protocols. *IEEE Access* **2018**, *6*, 66210–66222. [CrossRef]
28. Fantazzini, D.; Kolodin, N. Does the hashrate affect the bitcoin price? *J. Risk Financ. Manag.* **2020**, *13*, 263. [CrossRef]
29. Eskandari, S.; Leoutsarakos, A.; Mursch, T.; Clark, J. A first look at browser-based cryptojacking. In Proceedings of the 2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), London, UK, 24–26 April 2018; pp. 58–66.
30. Pastrana, S.; Thomas, D.R.; Hutchings, A.; Clayton, R. Crimebb: Enabling cybercrime research on underground forums at scale. In Proceedings of the 2018 World Wide Web Conference, Lyon, France, 23–27 April 2018; pp. 1845–1854.
31. Saad, M.; Khormali, A.; Mohaisen, A. End-to-end analysis of in-browser cryptojacking. *arXiv* **2018**, arXiv:1809.02152.
32. Schneier, B. *Secrets and Lies: Digital Security in a Networked World*; John Wiley & Sons: Hoboken, NJ, USA, 2015.
33. Seigen, M.; Jameson, T.; Nieminen, N.; Juarez, A. Cryptonight hash function. In *Cryptonote Standard 008*; 2013. Available online: https://github.com/deepwn/deepMiner/blob/master/CryptoNight.txt (accessed on 20 April 2021).
34. Rauchberger, J.; Schrittwieser, S.; Dam, T.; Luh, R.; Buhov, D.; Pötzelsberger, G.; Kim, H. The other side of the coin: A framework for detecting and analyzing web-based cryptocurrency mining campaigns. In Proceedings of the 13th International Conference on Availability, Reliability and Security, Hamburg Germany, 27–30 August 2018; pp. 1–10.
35. Varlioglu, S.; Gonen, B.; Ozer, M.; Bastug, M. Is cryptojacking dead after coinhive shutdown? In Proceedings of the 2020 3rd International Conference on Information and Computer Technologies (ICICT), San Jose, CA, USA, 9–12 March 2020; pp. 385–389.
36. 2021 Webroot BrightCloud Threat Report. Available online: https://i.crn.com/sites/default/files/ckfinderimages/userfiles/images/crn/custom/2021/2021_Webroot_BrightCloud_Threat_Report.pdf (accessed on 20 April 2021).

37. Aziz, A.B.A.; Ngah, S.B.; Dun, Y.T.; Bee, T.F. Coinhive's monero drive-by crypto-jacking. In *IOP Conference Series: Materials Science and Engineering*; IOP Publishing: Bristol, UK, 2020; p. 012065.
38. Berecz, G.; Czibula, I. Hunting Traits for Cryptojackers. In Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, (SECRYPT), Prague, Czech Republic, 26–28 July 2019; pp. 386–393.
39. Razali, M.A.; Shariff, S.M. Cmblock: In-browser detection and prevention cryptojacking tool using blacklist and behavior-based detection method. In Proceedings of the International Visual Informatics Conference, Kebangsaan, Malaysia, 19–21 November 2019; pp. 404–414.
40. Hong, G.; Yang, Z.; Yang, S.; Zhang, L.; Nan, Y.; Zhang, Z.; Duan, H. How you get shot in the back: A systematical study about cryptojacking in the real world. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1701–1713.
41. Alexa. The Top 500 Sites on the Web. Available online: https://www.alexa.com/topsites (accessed on 20 April 2021).
42. Burgess, J.; Carlin, D.; O'Kane, P.; Sezer, S. Manic: Multi-step assessment for crypto-miners. In Proceedings of the 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), Oxford, UK, 3–4 June 2019; pp. 1–8.
43. Carlin, D.; O'kane, P.; Sezer, S.; Burgess, J. Detecting cryptomining using dynamic analysis. In Proceedings of the 2018 16th Annual Conference on Privacy, Security and Trust (PST), Belfast, UK, 28–30 August 2018; pp. 1–6.
44. Wang, W.; Ferrell, B.; Xu, X.; Hamlen, K.W.; Hao, S. Seismic: Secure in-lined script monitors for interrupting cryptojacks. In Proceedings of the 23rd European Symposium on Research in Computer Security (ESORICS), Barcelona, Spain, 3–7 September 2018; pp. 122–142.
45. Gonzalez-Cuautle, D.; Hernandez-Suarez, A.; Sanchez-Perez, G.; Toscano-Medina, L.K.; Portillo-Portillo, J.; Olivares-Mercado, J.; Perez-Meana, H.M.; Sandoval-Orozco, A.L. Synthetic Minority Oversampling Technique for Optimizing Classification Tasks in Botnet and Intrusion-Detection-System Datasets. *Appl. Sci.* **2020**, *10*, 794. [CrossRef]
46. Hwang, J.; Kim, J.; Lee, S.; Kim, K. Two-stage ransomware detection using dynamic analysis and machine learning techniques. *Wirel. Pers. Commun.* **2020**, *112*, 2597–2609. [CrossRef]
47. Liu, J.; Zhao, Z.; Cui, X.; Wang, Z.; Liu, Q. A novel approach for detecting browser-based silent miner. In Proceedings of the 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC), Guangzhou, China, 18–21 June 2018; pp. 490–497.
48. Gomes, F.; Correia, M. Cryptojacking Detection with CPU Usage Metrics. In Proceedings of the 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 24–27 November 2020; pp. 1–10.
49. Kumar, C.O.; Bhama, P.R.S. Detecting and confronting flash attacks from IoT botnets. *J. Supercomput.* **2019**, *75*, 8312–8338. [CrossRef]
50. Morales-Molina, C.D.; Hernandez-Suarez, A.; Sanchez-Perez, G.; Toscano-Medina, L.K.; Perez-Meana, H.; Olivares-Mercado, J.; Portillo-Portillo, J.; Sanchez, V.; Garcia-Villalba, L.J. A Dense Neural Network Approach for Detecting Clone ID Attacks on the RPL Protocol of the IoT. *Sensors* **2021**, *21*, 3173. [CrossRef] [PubMed]
51. Xu, Y.; Zhou, Y.; Sekula, P.; Ding, L. Machine learning in construction: From shallow to deep learning. *Dev. Built Environ.* **2021**, 6, 100045. [CrossRef]
52. NoCoin 2018. Available online: https://github.com/keraf/NoCoin (accessed on 20 April 2021).
53. Raghavendra, S. Introduction to Selenium. In *Python Testing with Selenium*; Apress: Berkeley, CA, USA, 2021; pp. 1–14.
54. Orkphol, K.; Yang, W. Word Sense Disambiguation Using Cosine Similarity Collaborates with Word2vec and WordNet. *Future Internet* **2019**, *11*, *114*. [CrossRef]
55. Public WWW. 2020. Available online: https://publicwww.com/ (accessed on 20 April 2021).
56. Lachin, J.M. Fallacies of last observation carried forward analyses. *Clin. Trials J.* **2016**, *13*, 161–168. [CrossRef] [PubMed]
57. Al Shalabi, L.; Shaaban, Z.; Kasasbeh, B. Data mining: A preprocessing engine. *J. Comput. Sci.* **2006**, *2*, 735–739. [CrossRef]
58. Yousefi-Azar, M.; Varadharajan, V.; Hamey, L.; Tupakula, U. Autoencoder-based feature learning for cyber security applications. In Proceedings of the 2017 International joint conference on neural networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 3854–3861.
59. Siddiqui, A.J.; Boukerche, A. Adaptive ensembles of autoencoders for unsupervised IoT network intrusion detection. *Computing* **2021**, *3*, 1209–1232. [CrossRef]
60. Yang, H.; Zeng, R.; Xu, G.; Zhang, L. A network security situation assessment method based on adversarial deep learning. *Appl. Soft Comput.* **2021**, *102*, 107096. [CrossRef]
61. Albasir, A.; Hu, Q.; Naik, K.; Naik, N. Unsupervised detection of security threats in cyberphysical system and IoT devices based on power fingerprints and RBM autoencoders. *J. Surveillance Secur. Saf.* **2021**, *2*, 1–25. [CrossRef]
62. Zhou, C.; Paffenroth, R.C. Anomaly detection with robust deep autoencoders. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 665–674.
63. Rezvy, S.; Petridis, M.; Lasebae, A.; Zebin, T. Intrusion detection and classification with autoencoded deep neural network. In Proceedings of the International Conference on Security for Information Technology and Communications, Bucharest, Romania, 16–22 October 2018; Springer: Cham, Switzerland, 2018; pp. 142–156.
64. Song, Y.; Hyun, S.; Cheong, Y.G. Analysis of Autoencoders for Network Intrusion Detection. *Sensors* **2021**, *21*, 4294. [CrossRef]
65. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.

66. Cichy, R.M.; Kaiser, D. Deep neural networks as scientific models. *Trends Cogn. Sci.* **2019**, *23*, 305–317. [CrossRef] [PubMed]
67. Ullah, F.; Naeem, H.; Jabbar, S.; Khalid, S.; Latif, M.A.; Al-Turjman, F.; Mostarda, L. Cyber security threats detection in internet of things using deep learning approach. *IEEE Access* **2019**, *7*, 124379–124389. [CrossRef]