Cybersecurity

RESEARCH Open Access

# LSTM RNN: detecting exploit kits using redirection chain sequences

Jonah Burgess*, Philip O'Kane, Sakir Sezer and Domhnall Carlin

**Abstract**

While consumers use the web to perform routine activities, they are under the constant threat of attack from malicious websites. Even when visiting 'trusted' sites, there is always a risk that site is compromised, and, hosting a malicious script. In this scenario, the injected script would typically force the victim's browser to undergo a series of redirects before reaching an attacker-controlled domain, which, delivers the actual malware. Although these malicious redirection chains aim to frustrate detection and analysis efforts, they could be used to help identify web-based attacks. Building upon previous work, this paper presents the first known application of a Long Short-Term Memory (LSTM) network to detect Exploit Kit (EK) traffic, utilising the structure of HTTP redirects. Samples are processed as sequences, where each timestep represents a redirect and contains a unique combination of 48 features. The experiment is conducted using a ground-truth dataset of 1279 EK and 5910 benign redirection chains. Hyper-parameters are tuned via K-fold cross-validation (5f-CV), with the optimal configuration achieving an F1 score of 0.9878 against the unseen test set. Furthermore, we compare the results of isolated feature categories to assess their importance.

**Keywords:** Exploit kits, Malware, LSTM

## Introduction

Despite decades of research, browser and web-based threats remain a critical attack vector. In 2019, researchers identified an increasing trend of web exploitation, including a 460% increase in PowerShell attacks used to compromise servers, spawn remote webshells, deliver malware and establish botnets. CryptoJacking attacks increased by 29%, and, CookieMiner was discovered; a new malware family which targets Apple users, automating the theft of cryptocurrency site credentials (McAfee 2019).

EK attacks have declined in recent years due to fear of arrest, stronger offence by security vendors, increased security features in operating systems and browsers, the use of ad-blockers, more robust patching etc (Ma 2018). However, researchers registered nine active EKs in 2019. Although these EKs did not include new, zero-day vulnerabilities, they introduced innovative evasion techniques e.g. UnderMiner added steganography to provide security through obscurity (MalwareBytes 2020).

The web is also plagued by ClickJacking, FormJacking, FakeUpdaters, web-skimmers, malvertising and other browser-based attacks. Although these attacks differ in methodology, there may be universal features which can be extracted and used for detection. When combined with content-based features, the structure of HTTP redirections performed by websites could provide enough context to accurately classify malicious behaviour via machine learning (ML), which is the focus on this work.

The remainder of this paper is organised as follows. "Background" section provides a background into HTTP redirections, Exploit Kits and LSTM networks. "Related work" section surveys related works. In "Experiment" section, we describe our experimental methodology, present and evaluate our results, and, discuss system limitations. We conclude in "Conclusion" section and suggest areas of future work.

*Correspondence: jburgess03@qub.ac.uk
Centre for Secure Information Technologies (CSIT), Queen's University Belfast, Northern Ireland Science Park, Queen's Road, Belfast BT3 9DT, UK

## Background

### HTTP redirections

A HTTP redirection is an automatic transfer from one URL to another, without user interaction. Redirections broadly fall into two categories; header and content-based. Header-based redirects are usually achieved by populating the 'referrer' or 'location' header field and may produce a 30x HTTP code. These redirects are easy to map because their occurrence cannot be hidden or obfuscated in the network traffic.

Content-based redirects are launched from JavaScript (JS), HTML tags or iFrames, and, cannot be mapped using HTTP headers. Furthermore, these methods can be used in conjunction with obfuscation techniques that hinder the ability to trace redirections via static or manual analysis. A series of HTTP redirects is essentially a chain of URLs, sequenced by their time of access. Attackers use redirections to target victims with web-based malware while evading detection.

A common technique is to compromise a legitimate website and inject a script which redirects users to another domain. The injected script may go unnoticed for a period of time, especially if the redirect occurs silently. In the meantime, victims are redirected to a malicious site which may perform crypto-mining (Carlin et al. 2019) or attempt to exploit vulnerabilities (Kotov and Massacci 2013).

### Exploit kits

An EK is a malicious software package that can be used to automate the exploitation of computer systems. They operate on a subscription-based business model that drives innovation; authors protect their source-code, develop user-friendly interfaces, invest in new exploits, improve evasion/anti-analysis techniques, and, offer live customer support. The standard EK workflow can be broken down into five phases:

1. **Traffic Generation:** EK operators employ a variety of approaches to maximise traffic:

   - **Compromise Legitimate Websites:** inject malicious code which redirects to an EK.
   - **Malvertising Attacks:** place malicious code in an advertisement which will be displayed on a prominent, trusted website.
   - **Spam Email Campaigns:** trick users into opening a malicious link via phishing/spam.

   Search engine optimisation (SEO) poisoning is often used to provide a further boost to traffic.
2. **Redirections:** When a victim visits a compromised site or is served a malicious advertisement, their browser is redirected through a series of intermediate pages known as gates. This chain of redirects obscures the final URL that the victim will arrive at; the landing page.
3. **Fingerprinting:** When a victim arrives at the landing page, the fingerprinting process starts (may occur during the redirection phase too). Client/server-side code is used to profile the system and identify vulnerabilities.
4. **Exploitation:** If the EK identified a vulnerability for which it has a corresponding exploit, it now attempts to execute it. If multiple vulnerabilities were discovered, it may queue a series of exploits and execute each one until it successfully compromises the system.
5. **Payload Delivery:** Following successful exploitation, the payload is executed. Typically, the payload fetches a malicious binary, stores it on the victim machine and then executes it. The binary could be any variety of traditional malware e.g. ransomware or banking trojan.

Figure 1 outlines a common EK attack structure.

### Long short-term memory (LSTM)

Recurrent neural networks (RNNs) are a type of neural network (NN) that use internal memory to store contextual information. While a traditional NN takes a fixed length input maps it to an output, RNNs take a series of input and learn how inputs over time map to an output. These properties mean that RNNs are well suited to sequential and time-based problems where previous steps can be used to provide context and improve future predictions e.g. speech recognition, language translation, weather/stock market predictions.

Problems can arise with RNNs when the weight update procedure introduces weight changes so small that they have no effect (vanishing gradients) or so large that they create instability (exploding gradients). An LSTM network is a form of RNN which uses memory blocks to learn long-range dependencies across timesteps, addressing the gradient complications (Brownlee 2017).

LSTMs process sequences one timestep at a time, permitting variable length inputs and outputs. They can look back across approximately 100 times more timesteps than standard RNNs (Staudemeyer and Morris 2019). The layers in an LSTM network consist of sets of recurrently connected memory blocks. Each of these blocks contains one or more recurrently connected memory cells which have weight parameters for their input, output and internal state. The cells use three multiplicative gates to control their state and regulate information flow.

- **Forget:** What to forget from the previous cell.
- **Input:** What to learn from the previous cell.
- **Output:** What to pass on to the next cell.

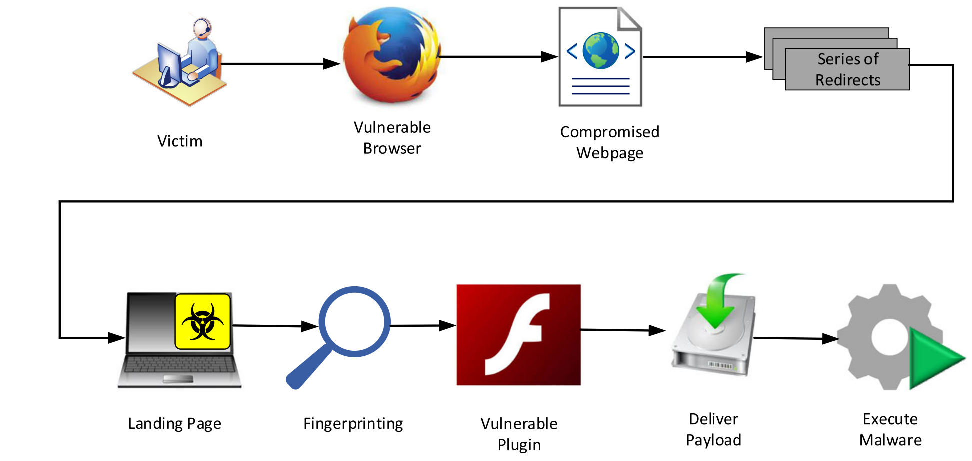

**Fig. 1** Typical EK attack structure

Figure 2 shows the structure of a LSTM memory cell and how gates control the cell state and govern information flow.

## Related work

Table 1 outlines the dataset, approach and results of related works which focus on malicious website detection by modelling the structure of redirection chains, or, using redirection-based features. The existing works have potential limitations or contrasting properties that distinguish it from this experiment.

Nikolaev et al. (2016) aims to detect a single malicious flow rather than chain of redirections, ignores content-based redirects and fails to properly validate dataset. Similarly, (Harnmetta and Ngamsuriyaroj 2018) focuses on individual network flows rather than the full sequence of flows that make up an EK attack. Singh and Goyal (2019) evaluates general web-based malware features for ML detection with no specific focus on EKs or the structure of malicious redirections, and, no explicit detection system is proposed.

The dataset in Süren et al. (2019) is limited (2016, spanning 4 families), and, attackers can update URL patterns using DGAs. Stringhini et al. (2013) uses a private dataset, and, fails to differentiate between header and content-based redirects. Li et al. (2014) relies on a clean version of the JS being available, doesn't cover all EK attacks, and, is easily bypassed by injecting malicious scripts in non-standard libraries or directly into webpage.

The small sample size in Matsunaka et al. (2014) may be problematic, and, URLs accessed within 2 seconds are labelled as related, but URLs accessed after user interaction, e.g. moving the mouse, labelled as initial entry points (potentially incorrectly). In Mekky et al. (2014), data is collected via browser rather than the logs typically available to organisations, and, the private dataset is labelled using an IDS which cannot detect unknown attacks.

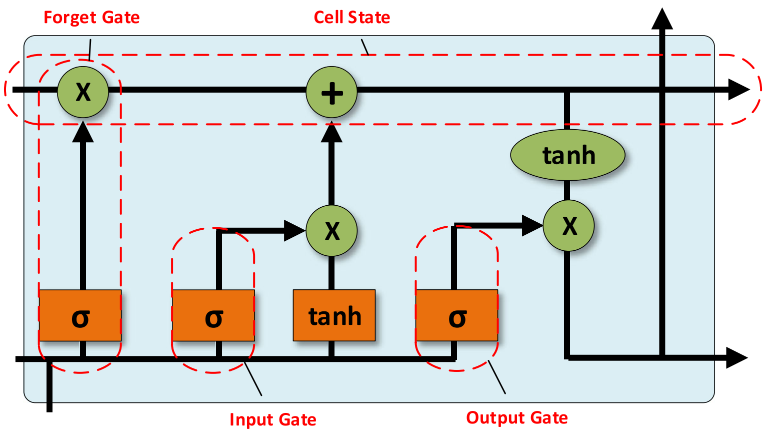

**Fig. 2** LSTM memory cell

**Table 1** Comparison of related works

| Ref | Dataset | Approach | Results |
|---|---|---|---|
| Nikolaev et al. (2016) | HTTP logs from 200+ networks over 6 months and PCAPs (Duncan 2020) over 3 months | Compares EK detection using 5 indicators (MIME type, structure, duration, repetition, browser agent) against RegEx only based detection | Average precision of 0.95 and recall of 0.92-0.95 using all 5 indicators |
| Harnmetta and Ngamsuriyaroj (2018) | 820 PCAPs (Duncan 2020) (2014-2016) | Applies Decision Tree classifier to content-based, interaction and connection-specific features extracted from the HTTP, DNS and Files logs produced by Zeek | Classified EK traffic with 0.99 accuracy, 0.92 precision and families with 0.82-0.99 accuracy, 0.8-0.99 precision |
| Singh and Goyal (2019) | Dataset extracted from 3496 malicious and 2907 benign websites (MalCrawler) | Determines importance of 25 different features for detecting malicious websites, according to accuracy and computational costs. Applies 10-fold cross-validation (CV) in WEKA using Naive Bayes and C4.5 classifiers | Identifies top 5 attributes of malicious sites; cloaking, use of iFrame, redirection, size of obfuscated code and pop-ups using Window.open() function |
| Süren et al. (2019) | 240 PCAPs (Duncan 2020) (2016) | Extracts 20 URL-based features from each domain in EK attack chain and compares ML algorithms | KNN, SVM, GBC achieved 0.958, 0.916 and 1.0 accuracy |
| Stringhini et al. (2013) | 5000 redirect chains from a large AV vendor (2012) | Builds redirection graphs by aggregating redirect chains from a collection of different users, and, extracts 28 features from 5 categories for SVM | Achieved F1 score of up to 0.881, depending on the range of features considered |
| Li et al. (2014) | Crawled Alexa top 1m domains and Microsoft's feed of malicious URLs over 4-6 weeks (2012) | Detects mass redirect-script injections by comparing suspicious JS files to their original versions. Based on the observation that redirection scripts are often quietly injected into legitimate JS libraries, whose unaltered code is publicly available | Produced detailed analysis of malicious JS/redirects and quantified the use obfuscation/evasion techniques |
| Matsunaka et al. (2014) | D3M 2013 dataset of 108 malicious websites (Marionette) | Uses monitoring sensors on the client-side (browser, web proxy and DNS), and, an analysis centre on the server-side to detect EK attacks. EXE downloads are classified as malicious if the URL is not present in previous HTTP headers or web content | Achieved 0% FPR with 24.2% FNR when tested against dataset of 108 URLs (33 malicious) |
| Mekky et al. (2014) | 15,000 malicious paths and 225,000 benign paths, provided by a large ISP (2011-2012) | Reconstructs user browsing activity into trees, representing time-based sessions, and, extracts 8 redirection-based features for use with a Decision Tree classifier | Extracted redirection trees with 0.965 accuracy, and, classified with precision and recall values of 0.9-0.98 |
| Takata et al. (2015) | Crawled 19,899 EK landing pages over 3 years (Marionette) | Applies program slicing to JS; executes each code segment and extracts URLs, even when cloaking prevents the execution of malicious JS branches | Extracted 30,000 new URLs compared to existing techniques |
| Nelms et al. (2015) | Dataset of 683 manually labelled, malicious download paths (164 EK instances) | Investigates browsing paths followed by users before an attack. WebWitness identifies a malicious download and traces back through HTTP requests, building a tree of redirects that led to the malware | Identified EKs with 0.9919 accuracy when tested against 48 EK samples using 10-fold CV |
| Taylor et al. (2016) | 688 million redirection trees, extracted from 3800 hours of traffic (2013-2014) | Builds web session trees (WST) and extracts URL-based features. Subtree similarity searches are performed against the WSTs to identify node-level and structural similarities with known malicious trees | Achieved 95% FPR against a dataset of 85 EK samples, and, identified 28 new EK instances during analysis |

**Table 1** Comparison of related works (*Continued*)

| Ref | Dataset | Approach | Results |
|---|---|---|---|
| Nagai et al. (2019) | D3M 2015 dataset of 256 malicious websites (Marionette) | Builds WSTs similar to (Taylor et al. 2016), but, aims to handle incomplete redirection data using time-based clustering. Focus is on WST construction rather than feature extraction | Average accuracy of 0.862 using 2-f CV. Scored higher on EKs families represented in both train and test sets |
| Takata et al. (2018) | 8467 JS samples from 20,272 malicious websites (2012-2016) | Compares redirection graphs from browsers running different JS implementations to identify structural differences resulting from evasive code | Discovered several new evasion techniques that abuse JS implementation differences |
| Shibahara et al. (2019) | Crawled 455,860 websites, 1.3% labelled as malicious or evasive (2016) | Graph mining approach to detect malicious sites, even if full chain of redirects cannot be extracted. 22 redirect, HTML and JS-based features obtained from each graph, evaluated with RF classifier | Achieved F1 score of 0.766 for sites hosting EK URLs, and, identified 143 more malicious sites than conventional systems |

Takata et al. (2015) exclusively focuses on identifying redirects in JS code, no other redirect types or website features are considered. The sample size in Nelms et al. (2015) is limited, and, it aims to classify different categories of web malware. Taylor et al. (2016) uses a private dataset extracted from 3800 hours of traffic, but, only contains 131 EK instances. Furthermore, potential content-based redirects are modelled using a 5 second threshold, without verification.

The small dataset used in Nagai et al. (2019) misses many high profile EKs (Angler, Nuclear, Neutrino, Rig, Fiesta), and, modelling redirects based on time alone is problematic. Redirection chains are mapped in Takata et al. (2018), but, content-based redirects are not considered. Shibahara et al. (2019) models redirections irrespective of occurrence, e.g. if URL is found in JS but wasn't accessed, it's still labelled as a redirect. Dataset distribution may also impact these results.

LSTM has been applied to traditional malware (Hwang et al. 2019) and Android malware detection (Vinayakumar et al. 2018). It has also been used to detect malicious websites using URL-based features (Liang et al. 2017) and JS bytecode sequences (Fang et al. 2018). To our knowledge, this paper presents the first use of an LSTM classifier to detect EKs, utilising the sequential structure of malicious redirection chains.

The LSTM detection method builds upon REdiREKT (Burgess et al. 2020), a system designed to map HTTP redirection chains observed in EK attacks and extract features for ML. By processing a unique combination of 9 redirection techniques, REdiREKT correctly extracted 96.52% of malicious domains from 1279 EK samples, spanning 28 families and 8 campaigns, and, only failed to extract 0.7% of malicious chains. A ground-truth dataset was produced during the experiment (see "Experiment" section).

REdiREKT (Burgess et al. 2020) serves as the first component of an EK detection system. This work advances the previous work and presents several new contributions; it is the first known application of a LSTM network to detect EK traffic, and using REdiREKT's new dataset, it better reflects up to date network traffic behaviours. The methods used to prepare the raw dataset for ML and build the LSTM classifier were carefully considered and are outlined in detail.

A range of hyper-parameters are explored and tuned using 5-fold CV to obtain the optimum model in terms of accuracy and computational cost, and, assess the fluctuation in results across different configurations. The final model is tested for environmental bias and feature categories importance are presented and discussed. The project code, dataset and results are publicly released to help further research in the field.

## Experiment

The experiment was conducted in 2020 on an Ubuntu 18.04 VM with an Intel i7-8700K CPU and 24GB RAM. The project is primarily written in Python3.

### Goals

1 Develop an LSTM model to detect EK attacks, utilising the sequential structure of redirections.
2 Apply the classifier to a ground-truth dataset.
3 Use K-fold CV to accurately tune hyper-parameters and select the best performing model.
4 Compare isolated feature category performance.
5 Assess environmental bias impact and discuss future work.

## Data extraction

In a previous experiment, REdiREKT (Burgess et al. 2020) was used to extract HTTP redirection chains from network traffic (PCAPs) using the Zeek IDS (Zeek 2020). A key goal of the experiment was to build a ground-truth dataset of EK and benign samples which could be used to assist future ML research. We use REdiREKT as our data collection and extraction system for this experiment. Further details of the methodology can be found in Burgess et al. (2020) but each sample is processed broadly as follows:

1 PCAP is processed by Zeek, generating a HTTP.log which is modified to include server-side headers. CONTENT.log is also generated via a custom Zeek script which uses RegEx to extract potential content-based redirects e.g. HTML/JS/iFrame/Base64/Concat.
2 Log entries are divided into sets according to their source IP, this enables tracking of multiple hosts.
3 If the time between two neighbouring entries is greater than 15 minutes (user-defined), they are split into temporal sessions.
4 Sessions are passed to the AnyTree module (c0fec0de 2020) to build redirection trees according to the redirects observed in the logs. Content-based redirects are verified first e.g. if an iFrame is found on a.com which points to b.com, we verify that b.com was accessed after a.com.
5 When all trees have been constructed, any single node trees (no redirects) that exist are compared against the leaf nodes in each tree. A 'subdomain' redirect is added if they have the same domain but different subdomains, and, were accessed with 60 seconds of each other.
6 Redirection chains are extracted from the trees. A chain is defined as a unique path from root node to one of its leaf nodes, excluding the root node's children which aren't on the direct path to the leaf node which are unlikely to be attacker-controlled.
7 The structure of each chain is exported to JSON.
8 48 features are extracted from each node, in each chain, and, are stored as rows in CSV file, uniquely indexed by the sample name, chain number and redirect number.

Time-based thresholds were selected pragmatically based on initial observations from our dataset and can be easily defined at runtime, allowing their impact to be assessed in different environments. Table 2 provides examples of the types of redirections processed by REdiREKT. The system is designed to overcome basic obfuscation techniques such as whitespace randomisation, case sensitivity and foreign characters.

Figure 3 shows a redirect tree and the four chains extracted from it. A chain from benign.com to x.sploit is extracted without modelling the siblings of 1.mal (children of benign.com not on the path to x.sploit). However, 2.mal is included in the chain, despite not being on the direct path to x.sploit. This is because, in an attack, 1.mal would be attacker-controlled and all redirects spawned from here are inherently malicious.

Figure 4 provides a high level overview of REdiREKT and Table 3 outlines the features which are currently extracted. Note, the initial feature-set is based on a combination of manual analysis of EK attacks and previous web-based malware research. DNS/file-based features will also be considered in future.

## Data preparation

Data produced by REdiREKT requires some preprocessing before it can be used for ML:

1 CSV dataset is loaded using Pandas.
2 Top level domain (TLD) is converted into category codes.
3 Features are scaled between 0-1 by MinMaxScaler.
4 Padding is used to ensure equal length sequences.
5 Dataset is reshaped into a 3D array (no_of_samples * max_nodes * features_per_node).

**Table 2** Redirection types

| Category | Type | Example |
|---|---|---|
| Header | Referrer | `referrer` field holds source, and `host` field holds destination URL |
| Header | Location | `host` field holds source, and `location` field holds destination URL |
| Content | HTML | `http-equiv="Refresh" url="< url>"` and `form\|a\|p\|img src="< url>"` |
| Content | JavaScript | `window\|document(.location\|.open)?.href\|hostname\|replace\|assign\|write` |
| Content | iFrame | `< iframe src="http://evil.com"></iframe>` |
| Content | Base64 | `window.cback('aHR0cDovL2V2aWwuY29tL2V4cGxvaXQvZXhwbG9pdC5waHA=');` |
| Content | Concatenation | `var a = "http://" + 'evil' + ".com"; window.href=a;` |
| Content | Unknown | URL found in page content and subsequently visited, no verifiable source |
| Relational | Subdomain | URL shares domain with recently accessed URL, no other identifiable source |

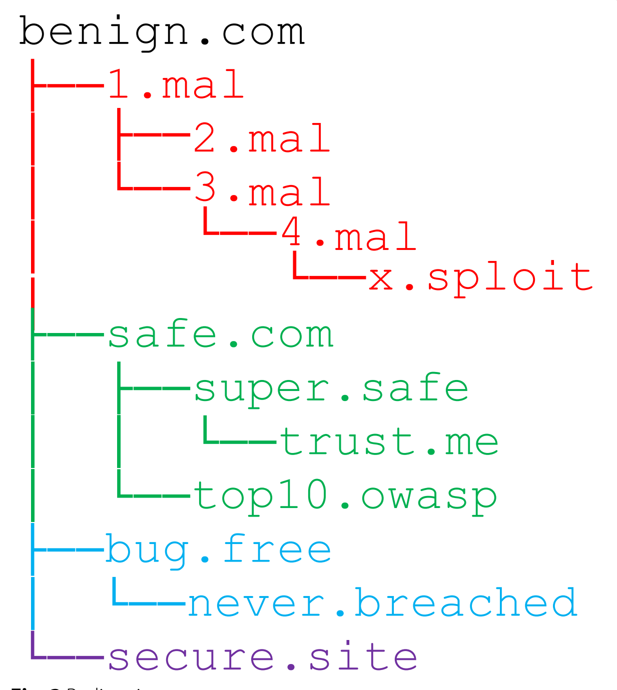

**Fig. 3** Redirection tree

**LSTM classifier**

An LSTM binary classification model is created to utilise the sequential structure of EK attacks and aid detection efforts. The classifer is developed using Jupyter Notebook and imports TensorFlow 1.14, Keras 2.3.1 and Scikit-Learn 0.22.2.

***Build model***

The model building function creates a new sequential model and adds a masking layer, with a mask value of zero. The masking layer ensures that all padded timesteps (consisting solely of zeroes) are ignored by the model. Next, X LSTM layers are added to the model consisting of Y neurons with a dropout value applied between each layer.

If X is greater than one, then return_sequences is set to True to ensure all the relevant data is passed to the next layer. The final LSTM layer will not have return_sequences enabled. A dense output layer with a single neuron and sigmoid activation function are added; they are designed for binary classification problems and allow the model to make 0-1 predictions.

Finally, the model is compiled using binary_crossentropy; the default loss function used for binary classification problems. Other common loss functions include hinge loss and squared hinge loss but they are primarily associated with SVM models. The adam optimiser is selected as it is known to perform well in most practical applications with minimal changes to hyper-parameters.

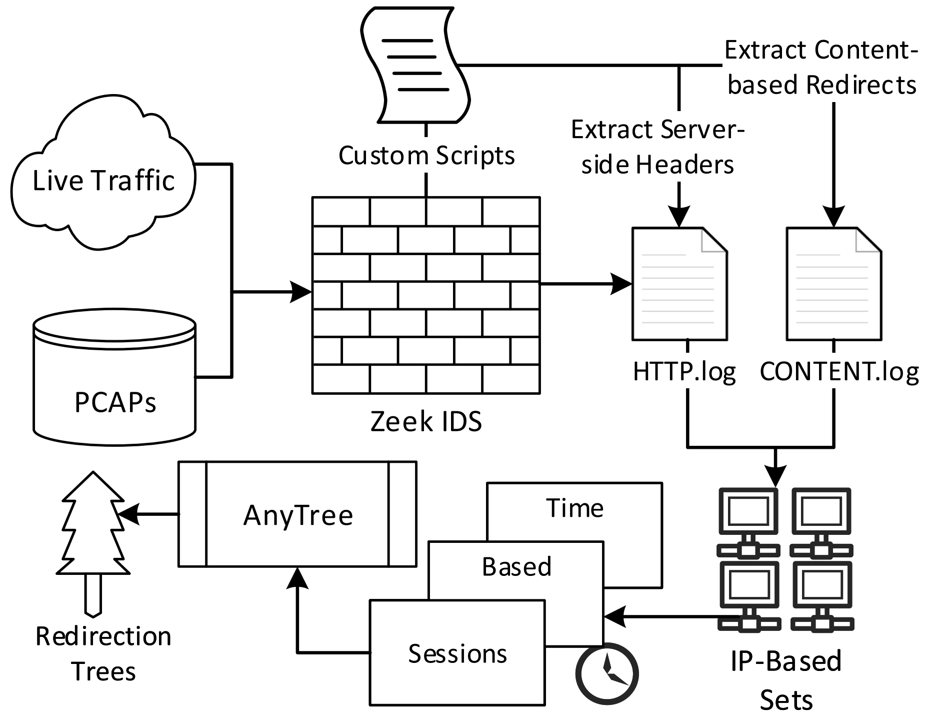

**Fig. 4** REdiREKT flowchart

**Table 3** Node-based features

| Feature | Description |
|---|---|
| Redirect | |
| Number | Index of node within chain |
| Depth | Depth of node within chain |
| Time | Time between redirections |
| Referrer | No. of 'Referrer' redirects |
| Location | No. of 'Location' redirects |
| HTML | No. of 'HTML' redirects |
| JS | No. of 'JS' redirects |
| iFrame | No. of 'iFrame' redirects |
| Subdomain | No. of 'Subdomain' redirects |
| Concatenation | No. of 'Concat' redirects |
| Base64 | No. of 'Base64' redirects |
| Unknown | No. of 'Unknown' redirects |
| URL | |
| Standard Port | Use of default HTTP(S) port |
| Is IP | Domain is an IP address |
| Domain Length | Length of the domain name |
| Domain Entropy | Entropy of the domain name |
| URI Length | Avg URI length |
| URI Entropy | Avg URI entropy |
| URI Slash | Avg/Total slashes ('/') |
| URI Amp | Avg/Total ampersands ('&') |
| URI Dash | Avg/Total dashes ('-') |
| URI Plus | Avg/Total pluses ('+') |
| TLD | Top-level domain |
| Content | |
| Requests | No. of HTTP requests |
| Response | Avg/Total size of responses |
| Shockwave | Avg/Total Shockwave bytes |
| Executable | Avg/Total EXE bytes |
| Java | Avg/Total Java bytes |
| Silverlight | Avg/Total Silverlight bytes |
| JavaScript | Avg/Total JavaScript bytes |
| XML | Avg/Total XML bytes |
| ZIP | Avg/Total ZIP bytes |
| Image | Avg/Total Image bytes |
| HTML | Avg/Total HTML bytes |

### Cross validation

CV is commonly used to assess the performance of ML models, especially when the dataset is limited. K-fold CV splits the dataset into K partitions, trains on K-1 partitions and tests on the remaining K partition. This process repeats K times until all samples have had a chance to appear in the test set, eliminating any bias introduced by the train:test dataset distribution. The final performance metric is calculated by averaging the scores for the K-folds.

In order to accurately select the optimal hyper-parameters for a model, the training set must be further broken down to include a validation set (train:val). The validation set is used to identify the best performing hyper-parameters, which, are then used in the final model to be evaluated against the test set (via K-fold CV). However, this method is susceptible to potential bias introduced by the train:val dataset distribution.

Nested CV can address this issue by applying multiple layers of K-fold CV. An inner loop applies K-fold CV to select hyper-parameters (train:val) and an outer loop applies K-fold CV to evaluate the model (train:test). Nested CV represents the highest standard of ML model selection and evaluation, but, it is computationally expensive and not always feasible.

Although the nested CV technique was initially implemented, it was ultimately discarded due to time constraints. Instead, the dataset is split into a train:test (80:20) set which remains fixed throughout the experiment. 5-fold CV splits the training set (train:val) during hyper-parameter tuning and the best performing model is evaluated with the unseen test set.

### Hyper-parameter tuning

Neural networks allow a range of hyper-parameters to be configured which determine the network structure and training properties. The configuration is not an exact science, and, although some formulas for calculating parameters exist, selection is often a trial and error-based approach. K-fold CV can be used to improve accuracy but it is computationally expensive.

The hyper-parameters listed below were tuned using 5-fold CV. Note that the network training time rises exponentially as the number of layers, neurons and epochs increase. LSTM networks allow many more hyper-parameters to be adjusted which were not tuned during this experiment due to time constraints. The default Keras values were used for learning rate (0.01), batch size (32) and any other hyper-parameters which are not described below.

- **Layers:** Number of hidden layers in network.
- **Neurons:** Number of hidden neurons in a layer.
- **Epochs:** Number of times that all samples in the training data are shown to the network and weights are updated.
- **Dropout:** Helps to avoid overfitting; dropout randomly skips neurons during training, increasing the robustness of the network.

Initially, GridSearchCV was used to perform an exhaustive, linear search over a grid of specified hyper-parameters, allowing the impact of incrementing each parameter to be visualised. Ultimately, RandomizedSearchCV was better suited for the final model selection as it allows the number of random iterations to be predefined, and, improves performance.

The best scoring model identified during the search is refit against the entire dataset (train:val) before testing. GridSearchCV and RandomizedSearchCV both perform data shuffling and use StratifiedKFold to ensure that each partition preserves the distribution of malicious and benign samples. The random_state values are fixed to ensure that model comparisons are accurate and experiments are reproducible.

## Dataset

The dataset used for this experiment was generated by processing PCAPs with REdiREKT (Burgess et al. 2020) (summarised in section B). The techniques used to build and verify the ground-truth dataset are broadly outlined here.

### *Malicious*

Malicious samples were collected from malware-traffic-analysis.com (Duncan 2020) and broadanalysis.com (Analysis 2020). Each PCAP is coupled with a technical report describing the malware e.g. family, campaign, exploit/malware type and associated domains. Only samples labelled as EK-traffic containing at least one redirect were considered. Some samples were excluded for a variety of reasons:

- Contained only post-infection traffic.
- Spam-based EK attack.
- Corrupt PCAP file.
- Author unable to provide password for archive.

Many samples were missing some of the initial traffic, e.g. the compromised host, often for privacy reasons. These PCAPs were not discarded, providing they contained at least one redirection. However, any samples missing the compromised host were clearly labelled, allowing statistics to be accurately updated and samples to be easily excluded from ML. The data collection process produced 1279 useable malicious samples from 2013-2019, spanning 28 EK families and 8 campaigns.

All samples were manually analysed using Wireshark alongside the accompanying technical reports. The output of this process was a series of JSON-based test cases which correctly map to the HTTP redirection chains observed in the PCAPs. When running REdiREKT against the malicious dataset, the extracted chains are compared to the corresponding test case. There are three possible test results for each sample:

- **Correct:** A chain matches the test case exactly.
- **Semi-correct:** A chain begins and ends with the correct domains, but, redirects are missing or ordered incorrectly.
- **Incorrect:** Failed to extract a chain beginning and ending with the correct domain.

Out of the 1279 malicious samples, 1172 (91.63%) malicious chains were correctly identified, 98 (7.66%) were semi-correctly identified, and 9 (0.7%) were incorrectly identified. In total, 3328 (96.52%) malicious domains were correctly extracted from 1279 malicious chains. Note that while semi-correct chains don't match the test case exactly, they may provide adequate data for ML as they still contain the EK domain which runs the exploit and delivers the payload.

### *Benign*

To generate the benign dataset, the Alexa top 10k websites were pre-filtered to remove URLs which were inactive, routed to the same domain (duplicates) or were HTTPS-based (currently unable to decrypt in Zeek - see "Conclusion" section), producing 1525 unique domains. Although this number may appear low, over 70% of the top 10k domains use HTTPS, which, is to be expected for high-profile websites. The remaining domains failed due to a variety of errors, e.g. connection refused, DNS lookup and timeout.

Next, each domain is queried using the VirusTotal API (VirusTotal 2019), and, excluded from the dataset if flagged by any AV vendors. 88/1525 domains were excluded for this reason. Selenium (Selenium 2019) is used to visit each domain while traffic is captured via TShark (Wireshark 2019). The system operates on Windows 10 using the native IE 11 browser to recreate the environment used to capture the malicious traffic as best possible.

If the page loads correctly, the Selenium driver attempts to close any generic GDPR/cookie-related pop-up windows and continues to capture traffic for 15 s. If the domain fails to load within 60 s, it is removed from the dataset. 37/1525 domains were excluded for this reason. Finally, the virtual environment is completely reset, ensuring a new session for each sample. This system produced 1400 PCAPs which were then processed through REdiREKT.

## Deployment

It is envisioned that the practical application of the LSTM model would occur at network level, using data extracted directly from networking equipment or HTTP logs. It is possible to capture data directly from the browser via an extension as seen in some existing works (Mekky et al. 2014). Using a browser extension offers a decentralised approach and has some key benefits e.g. easier to monitor HTTPS-based traffic and capture additional browser-related features.

However, there are also several problems with this method. Malicious JS could interfere with an in-browser detection system, poisoning data which is later used for ML training. Furthermore, in a successful EK infection the attacker will have complete control over the system and could easily disable the browser extension, clear logs and prevent reporting. Gaining control of an external networking device is significantly more challenging, even with full control of a system within the network.

The network-based data collection method was also chosen to improve accessibility. Most networks will have existing equipment to process HTTP traffic and store logs, whether that's a Firewall, IDS/IPS or standard HTTP Proxy. This means that many networks will be able to deploy the LSTM model against live traffic out of the box, and, even retroactively process stored logs to identify past infections.

The choice to focus on network traffic also ensures that historical EK attack data can be used for ML training. The malicious PCAPs which were gathered and processed for the experiment were captured from 2013-2019, on different computer systems and networks. Some of these PCAPs were generated using realistic honeypots, others were taken directly from the real-world attacks on corporate networks.

Unlike most traditional malware types, EK attacks cannot be easily recreated by executing a sample in a virtual machine. The EK needs to be active in order to recreate an attack, and, many of the popular families from the past decade have been taken down e.g. law enforcement takedown operations and EK developers ending their as-a-service offerings for a variety of reasons.

Even for the EKs that remain active, the ability to recreate attacks is greatly hindered by evasion techniques e.g. only target specific countries/OS/browser/software, only attempt infection once, VM/debugger aware etc. Therefore, it is important that existing data is utilised, and, the system can operate using data that is commonly recorded and stored by organisations. In future work, OS/browser-based features may be explored as an additional (possibly optional) data source.

## Metrics

Four common metrics are used to evaluate the ML model. True positives (TP) and true negatives (TN) refer to correctly classified samples (malicious and benign, respectively). False positives (FP) and false negatives (FN) refer to incorrectly classified samples (benign and malicious, respectively).

### *Accuracy*

Proportion of accurately classified samples to total number of samples. The accuracy metric can be misleading when dealing with an unbalanced dataset.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (1)$$

### *Precision*

Proportion of accurately classified malicious samples to total number of samples. A high precision score correlates to low FP rate (FPR).

$$precision = \frac{TP}{TP + FP} \quad (2)$$

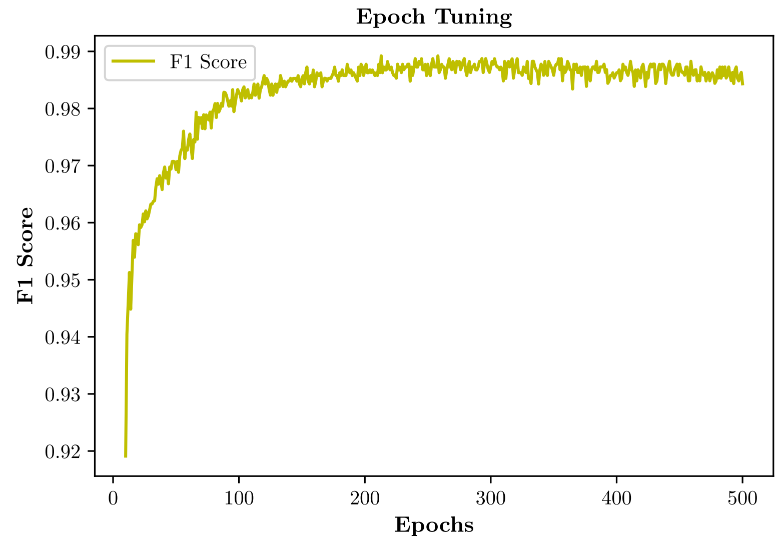

**Fig. 5** Testing 20-500 epochs on vanilla LSTM

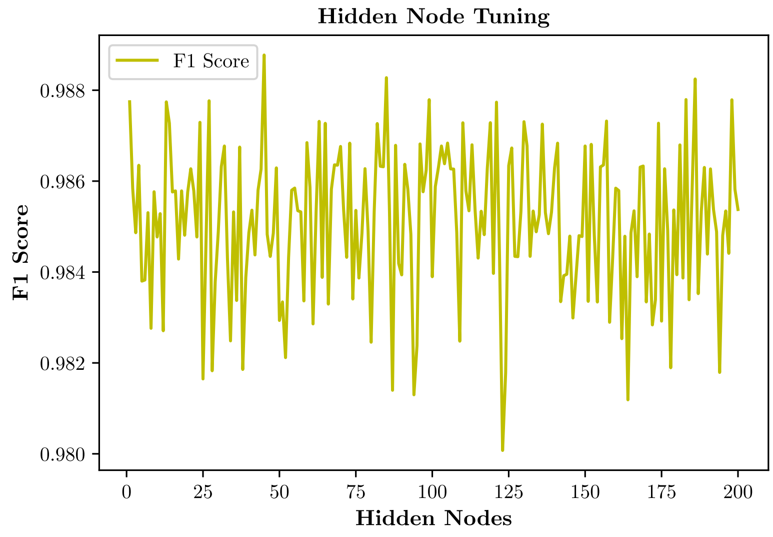

**Fig. 6** Testing 1-200 nodes (trained for 213 epochs)

### *Recall*

Proportion of accurately classified malicious samples to total number of malicious samples.

$$recall = \frac{TP}{TP + FN} \tag{3}$$

### *F1 score*

Harmonic mean of precision and recall. F1 is typically considered more valuable than accuracy, particularly when there is an irregular distribution of classes.

$$F1score = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{4}$$

## Results

### *Hyper-Parameter tuning*

5f-CV was applied during the hyper-parameter tuning process to identify the optimal range of parameters. Due to computational costs, a single iteration of 5f-CV was applied to the training set before retraining on the best performing parameters and testing the final model against the unseen test set. First, a vanilla model with a single hidden layer and neuron was constructed and trained for 10-500 epochs. Figure 5 shows that the performance plateaued after ~200 epochs, with the best performing epoch (213) achieving an F1 score of 0.989. If multiple models produce the same result, the parameter which incurs the least computational cost is selected.

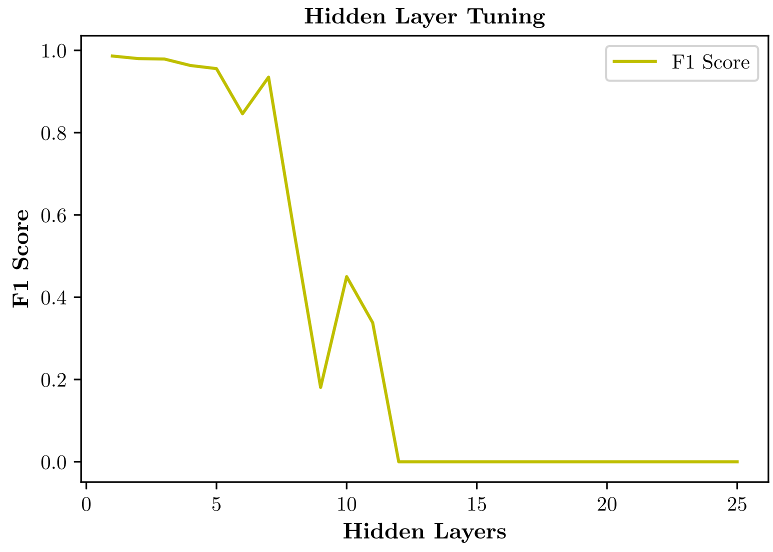

**Fig. 7** Testing 1-25 layers (trained for 213 epochs with 1 node)

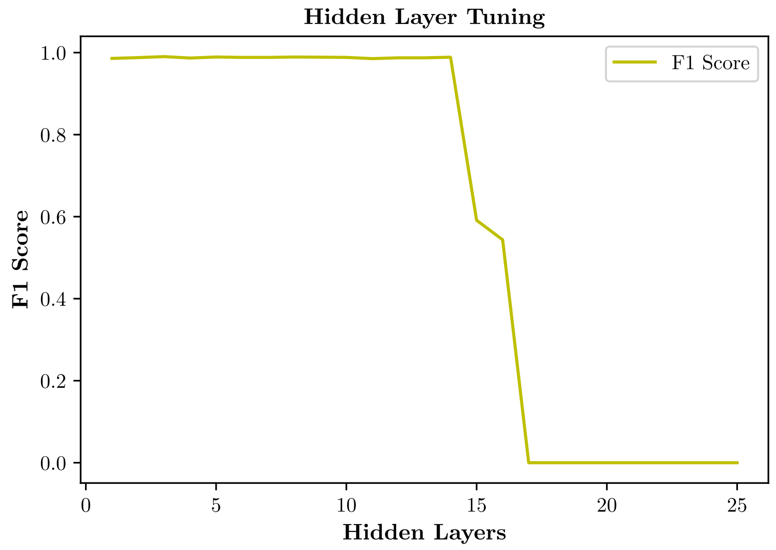

**Fig. 8** Testing 1-25 layers (trained for 213 epochs with 45 nodes)

The best performing epoch (213) was fixed throughout the remainder of the tuning process. This is not ideal because the optimum epoch is expected to change as other hyper-parameters are adjusted, but, it is a calculated trade off as computational costs will increase exponentially when tuning the number of layers and hidden units. This concern will be addressed later when RandomizedSearchCV is used to randomly select and test different hyper-parameter configurations, without introducing such significant resource costs.

Figure 6 shows the model performance across 1-200 hidden units with the optimal number of hidden units (45) achieving an F1 score of 0.989. Although the results appear to fluctuate significantly due to the graph scale, the standard deviation of 0.00161 is within the expected variation threshold for LSTM training. Figures 7 and 8 show the model performance across 1-25 layers with 1 hidden unit and 45 hidden units, respectively. The highest scores are 0.986 (1 layer) and 0.99 (3 layers), respectively. Note, the F1 score rapidly declines to zero in both cases where the number of layers is too high.

The drop was observed consistently across several tests using similar configurations and resulted from the sharp decline in precision and recall, while the accuracy remained relatively high. Keras applies dropout at the output gate between layers, randomly dropping neurons which helps prevent overfitting. Figure 9 shows the impact of the dropout hyper-parameter on the model

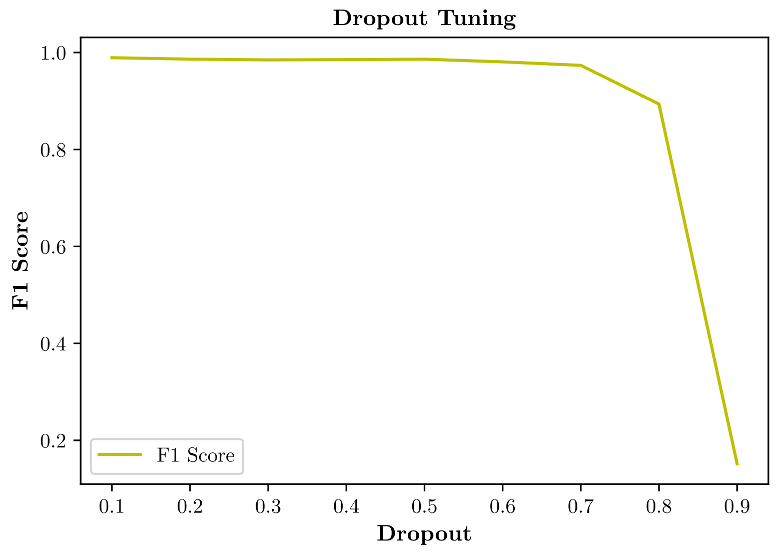

**Fig. 9** Testing 0.1-0.9 dropout (trained for 213 epochs, with 45 nodes and 1 layers)

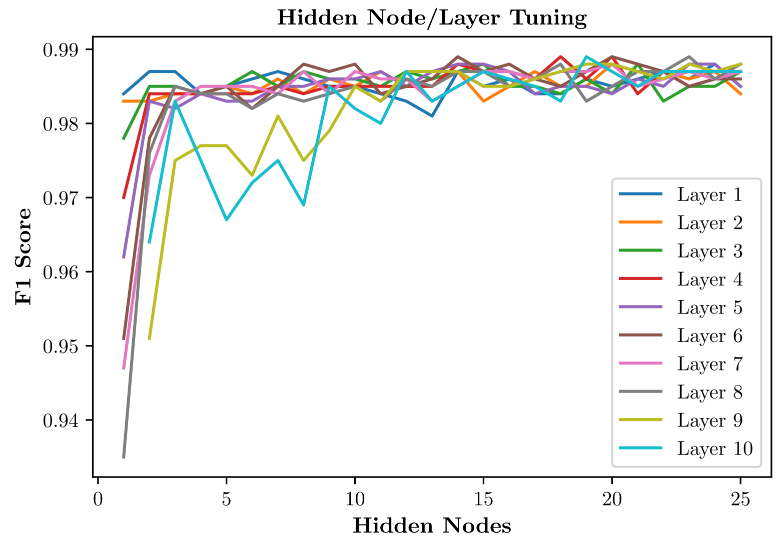

**Fig. 10** Testing 1-25 nodes across 1-10 layers (trained for 267 epochs)

performance. The dropout range 0.1-0.9 was tested and remained fairly consistent between 0.1-0.7 before declining. The best performing value was 0.1, achieving an F1 score of 0.989.

Figure 10 shows a full hyper-parameter grid search over 1-25 hidden units using 1-10 hidden layers, 3/250 results are not visualised due to anomalous values (0.5-0.7) that introduce scaling issues in the graph. Additional hyper-parameter testing was performed but not documented in the previous section as it did not produce noteworthy results e.g. halving the number of hidden units in each layer similar to (Hwang et al. 2019), and, various configurations based on existing academic work.

### *Optimum model*

Using insights gained from the GridSearchCV hyper-parameter tuning process, RandomizedSearchCV was configured with a grid of suitable hyper-parameters and executed 15 times. On each run, 10 grid configurations were randomly selected and tuned with 5f-CV. The best performing model was then retrained against the entire dataset (train:val) and evaluated against the test set.

The best model used 40 hidden units and 6 layers, and, was trained for 298 epochs with a dropout value of 0.4. The results from this model are presented in Table 4. To determine the importance of different feature categories, new models were constructed using the same hyper-parameters and trained for each feature type; redirect, URL and content (see Table 3).

The highest scoring individual category was content, achieving a score comparable to ‘all’ features. This is because EKs must deliver exploit code in the required format for execution e.g. Shockwave, Java, Silverlight. The URL category performed well since EKs generate domains with DGAs and use distinctive query strings. The detection of malicious sites via URL-based features is an extensive field, with notable works focusing on EKs (Süren et al. 2019) and one contribution applying LSTM to detect DGA-based domains (Liang et al. 2017).

Although the redirect feature category scored the lowest individually as a category, it still achieved notable results. Furthermore, the LSTM model is utilising the structure of redirections by processing each node as a timestep in a sequence. Therefore, the contribution provided by the redirect-based features is not fully represented in Table 4.

**Table 4** Results by feature category

| Feature | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| All | 0.9958 | 0.9918 | 0.9837 | 0.9878 |
| Redirect | 0.9318 | 0.87 | 0.7073 | 0.7803 |
| URL | 0.9708 | 0.9397 | 0.8862 | 0.9121 |
| Content | 0.993 | 0.9836 | 0.9756 | 0.9796 |

**Table 5** Environmental bias check

| Feature | Condition | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| Content | Averages only | 0.993 | 1.0 | 0.9593 | 0.9793 |
| All | No redirect timing | 0.9944 | 0.9877 | 0.98 | 0.9837 |
| All | Equalised dataset | 0.9882 | 0.9881 | 0.9881 | 0.9881 |

### *Environmental bias*

Although we aimed to replicate environments as closely as possible, the malicious and benign samples were captured on different systems over various time periods. This variation in the malicious dataset could actually provide robustness as it represents website visits from multiple sources, over a ~7 year period. The data was also captured using different hardware, operating systems and internet connections, which, are likely to have been replaced or upgraded along the way.

Based on the understanding that the average internet connection quality and website sophistication has been on a continuous upward trend, we wanted to test for environmental bias between the malicious and benign datasets. Although some of the malicious samples were captured recently, the average network connection speed and website size across the dataset is significantly lower than the benign samples which were all captured in 2019-2020.

We tested the 'content' features without totals to account for the increase of website content size and 'all' features without the redirect timing features, to account for the increased performance of web servers and high speed internet. Since benign redirect chains are over-represented in the dataset ~6:1, we also equalised the dataset to contain 1270 samples of each class. The impact of these environmental bias checks were negligible, as seen in Table 5.

### *Performance*

The accuracy of a ML model comes at a performance cost, and, achieving the right balance can be difficult. Depending on the use case, a significant increase in computational requirements may be justified for a slight boost in accuracy. Training a vanilla LSTM for 213 epochs achieves an F1 score of 0.989 during validation with an average fit and test time of 266 s and 0.26 s, respectively. Introducing 40 hidden units and 6 layers, and, training for 298 epochs (optimal model) achieves the same F1 score during validation while increasing the fit time by 1839%. Figure 11 show how the time consumption of training increases as more hidden units and layers are added the model.

## Conclusion

This paper presented the first known application of a LSTM classifier to detect EK traffic, utilising the sequential structure of HTTP redirections. Using a ground-truth dataset of 1279 EK and 5910 benign redirection chains obtained from REdiREKT (Burgess et al. 2020), we optimised LSTM model hyper-parameters and achieved an F1 score of 0.9878 against the unseen test set. Finally, we assessed the contribution of individual feature categories and the overall performance of the model.

In order to process HTTPS traffic at network level, the SSL keys must be captured and used to decrypt the traffic. While this is fairly trivial to do with right level of access and authorisation, automating the process in REdiREKT/Zeek remains a task for future work. Note that although this functionality is desirable, only 0.18% of the 1279 EK samples collected from 2013-2019 used HTTPS.

Additional data sources could be explored in future work to identify new detection features. The Zeek DNS.log could be used identify the age and scheduled validity of domain names, e.g. if a domain was registered recently and has a short expiration, it's more likely to be malicious. The Zeek Files.log may also prove useful; if an executable file is downloaded by the final node in a chain, it is more likely to be an attack than a redirection chain that does not drop a binary.

Finally, expanding the ground-truth dataset is a key goal of future work. In particular, we would like to apply REdiREKT to other forms of web-based malware and identify any shared or distinct characteristics which could be

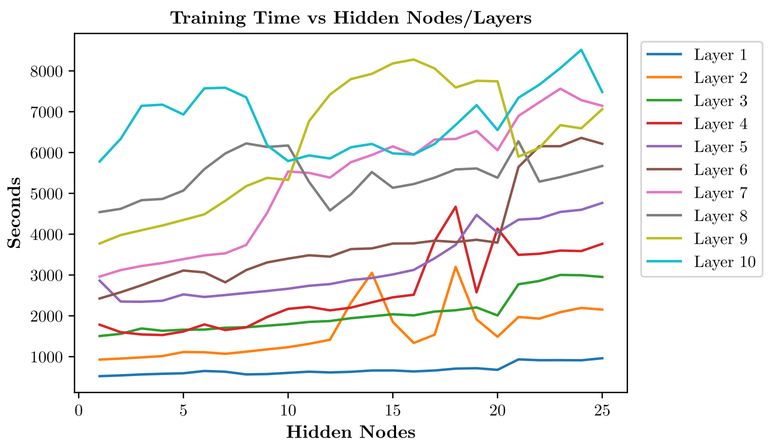

**Fig. 11** Time consumption vs hidden layers/nodes (trained for 267 epochs)

used to classify attacks, e.g. ClickJacking, CryptoJacking, FormJacking and FakeUpdaters. Dataset expansion is time consuming as it requires extensive manual analysis and labelling.

**Acknowledgements**
Not applicable.

**Authors' contributions**
Main author is Jonah Burgess, who carried out the experiment and authored the paper. The remaining authors helped with reviewing/proofreading/mentorship. The author(s) read and approved the final manuscript.

**Funding**
Not applicable.

**Availability of data and materials**
The dataset used to carry out this experiment has been published on GitHub and QUB Pure, along with the project code which captured and processed the data. Similarly, the code and dataset produced during this experiment will be published in the same locations.
All data and code will be published on QUB Pure / GitHub after publication. Code/data from the first part of this experiment (REdiREKT) is already published on Pure/GitHub. Any other data required available upon request.

## Declarations

**Ethics approval and consent to participate**
N/A

**Consent for publication**
N/A

**Competing interests**
N/A

Received: 12 February 2021 Accepted: 28 April 2021
Published online: 12 July 2021

**References**

Analysis B (2020) Broad Analysis. https://broadanalysis.com/. Accessed 7 May 2021
Brownlee J (2017) Long Short-term Memory Networks with Python: Develop Sequence Prediction Models with Deep Learning. Machine Learning Mastery
Burgess J, Carlin D, O'Kane P, Sezer S (2020) REdiREKT: Extracting Malicious Redirections from Exploit Kit Traffic. In: 2020 IEEE Conference on Communications and Network Security (CNS). IEEE
c0fec0de (2020) Python AnyTree Module. https://anytree.readthedocs.io/en/latest/. Accessed 7 May 2021
Carlin D, Burgess J, O'Kane P, Sezer S (2019) You could be mine (d): the rise of cryptojacking. IEEE Secur Priv 18(2):16–22
Duncan B (2020) Malware Traffic Analysis. https://www.malware-traffic-analysis.net/. Accessed 7 May 2021
Fang Y, Huang C, Liu L, Xue M (2018) Research on malicious javascript detection technology based on LSTM. IEEE Access 6:59118–59125
Harnmetta S, Ngamsuriyaroj S (2018) Classification of exploit-kit behaviors via machine learning approach. In: 2018 20th International Conference on Advanced Communication Technology (ICACT). IEEE. pp 468–473
Hwang R-H, Peng M-C, Nguyen V-L, Chang Y-L (2019) An LSTM-based deep learning approach for classifying malicious traffic at the packet level. Appl Sci 9(16):3414
Kotov V, Massacci F (2013) Anatomy of exploit kits. In: International Symposium on Engineering Secure Software and Systems. Springer, Berlin, Heidelberg. pp 181–196
Li Z, Alrwais S, Wang X, Alowaisheq E (2014) Hunting the red fox online: Understanding and detection of mass redirect-script injections. In: 2014 IEEE Symposium on Security and Privacy. IEEE. pp 3–18
Liang J, Zhao W, Ye W (2017) Anomaly-based web attack detection: a deep learning approach. In: Proceedings of the 2017 VI International Conference on Network, Communication and Computing. pp 80–85
Ma Z (2018) The decline of exploit kits as an exploitation strategy. https://www.doc.ic.ac.uk/~livshits/papers/theses/zicong_ma.pdf
MalwareBytes (2020) State of Malware Report 2020. https://resources.malwarebytes.com/files/2020/02/2020_State-of-Malware-Report.pdf. Accessed 7 May 2021
Matsunaka T, Kubota A, Kasama T (2014) An approach to detect drive-by download by observing the web page transition behaviors. In: 9th Asia Joint Conference on Information Security. IEEE. pp 19–25
McAfee (2019) McAfee Labs Threat Report: August 2019. https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-aug-2019.pdf. Accessed 7 May 2021
Mekky H, Torres R, Zhang Z-L, Saha S, Nucci A (2014) Detecting malicious http redirections using trees of user browsing activity. In: IEEE INFOCOM 2014-IEEE Conference on Computer Communications. IEEE. pp 1159–1167
Nagai T, Kamizono M, Shiraishi Y, Xia K, Mohri M, Takano Y, Morii M (2019) A malicious web site identification technique using web structure clustering. IEICE Trans Inf Syst 102(9):1665–1672
Nelms T, Perdisci R, Antonakakis M, Ahamad M (2015) Webwitness: Investigating, categorizing, and mitigating malware download paths. In: 24th {USENIX} Security Symposium 15. pp 1025–1040
Nikolaev I, Grill M, Valeros V (2016) Exploit kit website detection using http proxy logs. In: Proceedings of the Fifth International Conference on Network, Communication and Computing. ACM. pp 120–125
Selenium (2019) Selenium Browser Automation. https://www.seleniumhq.org/. Accessed 7 May 2021
Shibahara T, Takata Y, Akiyama M, Yagi T, Hato K, Murata M (2019) Evasive malicious website detection by leveraging redirection subgraph similarities. IEICE Trans Inf Syst 102(3):430–443
Singh A, Goyal N (2019) A comparison of machine learning attributes for detecting malicious websites. In: 2019 11th International Conference on Communication Systems & Networks (COMSNETS). IEEE. pp 352–358
Staudemeyer RC, Morris ER (2019) Understanding lstm–a tutorial into long short-term memory recurrent neural networks. arXiv preprint arXiv:1909.09586
Stringhini G, Kruegel C, Vigna G (2013) Shady paths: Leveraging surfing crowds to detect malicious web pages. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. ACM. pp 133–144
Süren E, Angin P, Baykal N (2019) I see EK: A lightweight technique to reveal exploit kit family by overall URL patterns of infection chains. Turk J Electr Eng Comput Sci 27(5):3713–3728
Takata Y, Akiyama M, Yagi T, Hariu T, Goto S (2015) Minespider: Extracting urls from environment-dependent drive-by download attacks. In: 2015 IEEE 39th Annual Computer Software and Applications Conference. IEEE Vol. 2. pp 444–449
Takata Y, Akiyama M, Yagi T, Hariu T, Ohkubo K, Goto S (2018) Identifying evasive code in malicious websites by analyzing redirection differences. IEICE Trans Inf Syst 101(11):2600–2611
Taylor T, Hu X, Wang T, Jang J, Stoecklin MP, Monrose F, Sailer R (2016) Detecting malicious exploit kits using tree-based similarity searches. In: Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy. ACM. pp 255–266
Vinayakumar R, Soman K, Poornachandran P, Sachin Kumar S (2018) Detecting android malware using long short-term memory (LSTM). J Intell Fuzzy Syst 34(3):1277–1288
VirusTotal (2019) VirusTotal. https://www.virustotal.com. Accessed 7 May 2021
Wireshark (2019) Wireshark - TShark. https://www.wireshark.org/docs/man-pages/tshark.html. Accessed 7 May 2021
Zeek (2020) The Zeek Network Security Monitor. https://www.zeek.org/. Accessed 7 May 2021

## Publisher's Note