# Numerical Encoding to Tame SQL Injection Attacks

Solomon Ogbomon Uwagbole
School of Computing
Edinburgh Napier University
Edinburgh, United Kingdom
s.uwagbole@napier.ac.uk

William J. Buchanan, Lu Fan
School of Computing
Edinburgh Napier University
Edinburgh, United Kingdom
b.buchanan;l.fan@napier.ac.uk

***Abstract*— Recent years have seen an astronomical rise in SQL Injection Attacks (SQLIAs) used to compromise the confidentiality, authentication and integrity of organisations' databases. Intruders becoming smarter in obfuscating web requests to evade detection combined with increasing volumes of web traffic from the Internet of Things (IoT), cloud-hosted and on-premise business applications have made it evident that the existing approaches of mostly static signature lack the ability to cope with novel signatures. A SQLIA detection and prevention solution can be achieved through exploring an alternative bio-inspired supervised learning approach that uses input of labelled dataset of numerical attributes in classifying true positives and negatives. We present in this paper a Numerical Encoding to Tame SQLIA (NETSQLIA) that implements a proof of concept for scalable numerical encoding of features to a dataset attributes with labelled class obtained from deep web traffic analysis. In the numerical attributes encoding: the model leverages proxy in the interception and decryption of web traffic. The intercepted web requests are then assembled for front-end SQL parsing and pattern matching by applying traditional Non-Deterministic Finite Automaton (NFA). This paper is intended for a technique of numerical attributes extraction of any size primed as an input dataset to an Artificial Neural Network (ANN) and statistical Machine Learning (ML) algorithms implemented using Two-Class Averaged Perceptron (TCAP) and Two-Class Logistic Regression (TCLR) respectively. This methodology then forms the subject of the empirical evaluation of the suitability of this model in the accurate classification of both legitimate web requests and SQLIA payloads.**

***Keywords—NETSQLIA; SQLIA; numerical attributes encoding; SQL Injection; SQLIA neurons***

## I. INTRODUCTION

A typical method used to pilfer confidential data is by SQLIA [1] with successful attacks leading to serious ramifications in ransom, extortion and loss of revenue to businesses. Security firewalls lock down ports and applications but often do little against malicious web requests stealthily concealed in legitimate web requests. This leaves most existing solutions, provided in Web Application Firewalls (WAF) [2] that rely on signature approaches to detect an attack, playing catch-up.

The methodology demonstrated in the proposed Numerical Encoding to Tame SQLIA (NETSQLIA) is an alternative bio-inspired method that combines static and dynamic analysis leveraging SQL parser and man-in-the-middle functionality of proxy for numerical attributes extraction from features (legitimate web requests, injection mechanisms and SQLIA types) in a web request (URL POST, GET and body contents). It is a self-contained model that combines collation of the numeric attributes and web services from a trained model built on Microsoft Azure Machine Learning (MAML) [3] studio for a continuous SQLIA detection; dropping suspect requests but forwarding legitimate web requests to the back-end database through custom Internet Information Services (IIS) web server.

There is a strong argument for a bio-inspired against existing signature based approach, not least because an intruder may exploit a SQLIA type with many signature variations in order to evade pattern matching e.g. a SQL injection tautological attack of 1=1 can also be written as 1 > 1, 'a'='a', etc. [4] to achieve the same attack.

These variations could create significant issues for signature based detection and prevention methods in recognising novel signatures that intruders continuously evolve. This paper explores these variations as the blue print for the random decimal attribute that augment these variations within a SQLIA type in the generation of large dataset items of any size. These random decimal attribute items can be given an identity by mapping to them patterns of these variations that exist within a SQLIA type.

NETSQLIA excludes new attack patterns not currently trained for as an outlier and thereby the model is able to carry-on with effective SQLIA detection and prevention.

Although the work presented in this paper may have all the hallmarks of a full SQLIA detection and prevention system, it is intended for a technique of labelled dataset extraction of any size primed as input to supervised learning models. Labelled dataset has attribute items which are classed as normal and suspect. This then forms an input dataset to a supervised model of ANN that is cross-validated with a statistical ML algorithm for the overall performance of Area Under Curve (AUC) which ANN comes on top with negligible difference of 0.02 (2%) against ML. The confusion matrix for the repeated ANN model training ranges between: AUC = 0.78 (78%) with accuracy of 0.911 (91.1%); and AUC = 0.912 (91.2%) with accuracy of 0.929 (92.9%).

We present here a proof of concept demonstration and validation of NETSQLIA implemented on MAML studio using training algorithms of TCAP [5] and TCLR [6]. This methodology then forms the subject of the empirical evaluation of the suitability of this model in the numerical attributes extraction and accurate classification of both legitimate web requests and SQLIA payloads.

The paper is laid out in six sections ending with a conclusion and future work summary. Section II covers background and theory and Section III is focused on related work; with Sections IV and V detailing NETSQLIA attributes encoding, evaluation and results.

## II. Background Theory

The approach presented in NETSQLIA intercepts web requests of any intent at the proxy for numerical attributes encoding of features based on legitimate web requests (valid requests), injection mechanisms and SQLIA types.

The injection mechanisms can be through: web page forms e.g. login screen; second-order injection by concealing a Trojan horse for a later date attack; exploiting web enabled server variables to gain access to back-end database; and through cookies that have stored state information then used to gain unauthorised access to the back-end database. NETSQLIA prevents second-order injections by enforcing in predefined patterns no special characters, spaces and encoding obfuscation in web form input being monitored. The use of proxy enables intercepted monitored web form input and cookies to be decrypted (if encrypted input is so desired) to be laid bare for thorough analysis.

Notable SQLIA types are techniques that can be employed in any combination to carry out an attack which includes: Tautology; Invalid/Logical Incorrect; Union; Piggy-backed; Store procedure; Time-based; and Alternate encoding obfuscation. Further reading on SQL types can be found in paper [7].

The long existence of the SQL injection problem has not provided a robust test dataset similar to the one used in the intrusion detection system of KDD Cup [8]. The few sample datasets that exist are usually project specific. These would normally contain unprocessed strings of repeating features of the variations that exist within SQLIA types as against pre-processed features of numeric attributes suited for artificial intelligence like ANN and ML presented here. The random decimal attribute introduced in NETSQLIA is aimed to provide a way to derive large dataset attributes encoded from features, which is often lacking in existing works.

With this attribute of random decimal values, it is possible to have a large dataset to simulate in ANN and ML a hypothetical scenario of tens of thousands to millions of SQLIAs that are likely with automated injection attack tools.

NETSQLIA is fully replicable as described here with .NET C# and R language background using open source software of fiddler proxy, SQL Script Dom Parser API, RegEx and MAML studio.

## III. Related work

Whilst it is acknowledged that there are existing works that share similarities in the overall scheme of SQLIA detection and prevention this does not extend to the input of numerical attributes extraction from features in-transition to a supervised learning model as proposed in this paper. This forms the context of reviewing a few selected related works.

A recent work [9] employing a hybrid of dynamic and static approach predicted code vulnerabilities against a supervised model which achieved 0.77 (77%) recall value. There was no AUC value provided in the paper to gauge the decency of the model in overall performance. In this proposed model, we have demonstrated a supervised learning model trained with primed datasets for a decent model of AUC of 0.78 (78%) with recall of 93.7%, and even at the high end of the scale with AUC of 0.912 (91.2%) with a recall value of 0.984 (98.4%).

SQLrand [10] is a related work that employs proxy and SQL parser but a signature approach. SQLProb [11] is also a proxy based approach but the proposed scheme addressed second-order attack which is not addressed in both approaches.

JDBC Checker [12] is a static approach that explores finite state automaton but unlike in this proposed model, it lacks proxy to backhaul web requests for thorough analysis.

AMNESIA [13] uses NFA static and dynamic pattern matching similar to the one used in the proposed scheme but the system solely relied on pattern matching and lacked method to address obfuscated requests aimed at evading pattern matching.

Some notable works are known for code validation but here web traffic is backhauled for deep pattern analysis at the proxy.

## IV. NETSQLIA

### A. Numerical attributes encoding

Fig.1 gives a high-level illustration of the three primary processes that highly depend on predefined NFA patterns implemented in RegEx. These patterns include: pre-defined patterns for expected legitimate requests; and patterns for outliers including known attack signatures ("?", "'", "OR", "1=1" and "--"), etc.

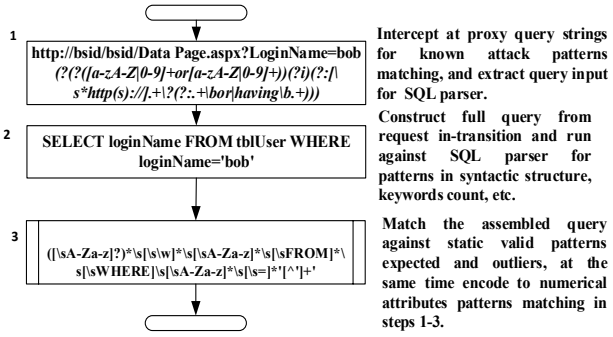


Fig. 1.Numerical attributes encoding

### B. Numerical Attributes interpretation

Table I illustrates the procedure of attributes extraction. If the static defined pattern *(p)* finds a match, then a number is assigned from a range of 1 to 9. The value of 9 being a legitimate (normal) web request value, and a value range of 1 to 8 being any of the SQLIA types not excluding second-order injection mechanisms. The threat threshold is the factor determining the value, which in this case means that a number equal to or greater than 9 will be deemed a legitimate request. It is imperative that all other numeric attributes dataset items stay above two decimal precisions *(e.g. 0.01...0.0$_n$)* meaning that the sum of all other attributes row items must be less than 1 as a value greater than 1 will shift the threat threshold value. If that is the choice, then the sum of all attributes row items must be calculated before deciding on the threat threshold value.

Matched patterns are further scored to determine *(d)* pattern match by implementing NFA backtracking [14] to cross-check if the criteria for features are being met. The scoring is modelled as follows: are the patterns being matched present in dynamic features (web requests); are patterns partly present in matched features; e.g. implementing backtracking for tautological attack will be a litmus test for escape character (‘), OR, patterns of 1=1

and its derivations. For simplicity, range values of 0.01 to 0.05 are assigned for a degree of precision of features in pattern matching.

The assignment of numeric values range is flexible so long as threat threshold is accounted for and the dataset attributes are normalised when applied as input vectors to ANN or ML.

TABLE I. FEATURES ENCODING PROCEDURE

**Getting the primary attributes of predictors or x variables**
*Get matched patterns (p) of features using NFA (RegEx)*
if features matched static defined patterns
assign a numeric value of 1 to 9
*Validate (v) matched pattern (p) using a method of NFA backtracking*
randomly assign a score 0.01 to 0.05
*Randomisation (r)*
randomly assigned a decimal value less than 0.1
computed against web requests total count
*Calculating the likeliness of being normal (n) threshold -risk factor*
*SUM (p, v, r)*, take the minimal value >= 9 as threshold
If n >=9
Then
Normal = 1
Else
Suspect = -1
**Calculating y –variables or what to predict**
If normal *(n)*
Then
0 1
Else
1 0

The labelled dataset attributes (predictors) are scaled down in this paper for simplicity but the approach can be replicated for as many attributes so desired that are attributed to SQLIA behaviour patterns.

The encoding procedure in Table 1 demonstrates how the numerical attributes detailed in Table II are obtained. The notation of the attributes of recognised pattern (sitype *p*) $\{w_0 \ldots w_n\}$; determinant *d* $\{w_0 \ldots w_n\}$ and subtype random risk *r* $\{r_0 \ldots r_n\}$ (the random decimal attribute scaled to *e.g. 0.01...0.0*$_n$). $w_0 \ldots w_n$ and $r_0 \ldots r_n$ are the dataset items (rows) values of the attributes detailed in Table II. These x-attributes SUM *(p, d, r)* are collated to obtain the likeliness of features being a risk factor that can either be -1 likeliness or 1 for remote likeliness. The y-variables or what to predict (commonly known as class) can be a possible 1 (1 0) for SQLIA or 0 (0 1) for normal as illustrated in Table I are inferred from risk factor *(r)*.

TABLE II. ENCODED NUMERIC DATA PRIMED FOR ANN AND ML

| Sitype *(p)* | Determinant *(d)* | Subtype random Risk *(r)* | Risk factor | class |
|---|---|---|---|---|
| $p\{w_0\}$ | $d\{w_0\}$ | $r_0$ | $x_o$ | $y_o$ |
| 1 | 0.03 | 0.0846 | -1 | *10* |
| 9 | 0.05 | 0.0482 | 1 | *01* |
| $p\{w_n\}$ | $d\{w_n\}$ | $r_n$ | $x_n$ | $y_n$ |

## C. Implementation

A high level overview of NETSQLIA is presented in this section along with Fig.2 which illustrates the four primary components of the complete schematic architecture of NETSQLIA in numerical attributes extraction from features that are fed as input to the supervised learning model. The stages are:

1) Dynamic web requests of features. A simulated live scenario can be fed from a static file containing features. Alternatively, the use of a unit testing tool e.g. selenium web testing tool can be used to simulate request automation. This method can also be used to automate the initial training phase.

2) Fiddler proxy [15] to backhaul web traffic for deep web request analysis that includes numerical attributes extraction from matched patterns before passing-on valid requests to an IIS.

3) Web requests payload decision engine collating the encoded numerical attributes to a labelled dataset of normal and threat payloads. SQLIA payloads are dropped based on this collation stage assisted by existing trained model exposed as web service in conjunction with form input validation.

4) Primed numerical attributes input to a better performing ANN (TCAP) that when cross-validated against ML (TCLR) produces a negligible difference, shown in Fig. 3 in Section V.

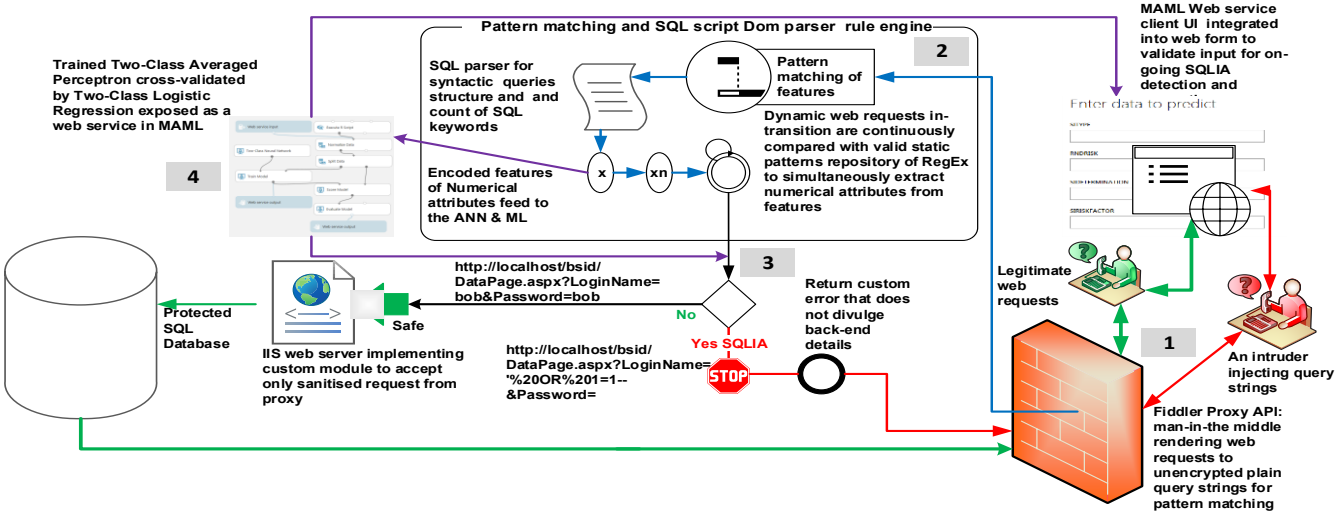


Fig.2. Proof of concept: attributes extraction, trained model web service for SQLIA detection and prevention

## V. Evaluation and Result

A total of 236 unique SQLIA features were collected across different hacking forums, websites and SQLIA tools. The novel approach described in numerical attributes extraction from features provides a method to generate a dataset of any size from the template patterns collected. There were two sets of datasets used which both contained four x-attributes (predictors) and two y-attributes (what to predict or labelled class) created for this experiment that were ran on MAML studio.

Dataset 1 contained 30,000 unadulterated items (rows) generated as described in Section IV, which the trained model gave an overfitting in all confusion matrix of a value of 1. This is expected for a primed labelled dataset. While dataset 2 had 50,000 dataset items that were randomly sorted to provide equal distribution of featured items. Dataset 2 contained a mixture of both 20,000 adulterated and 30,000 unadulterated dataset items. The adulteration part of dataset 2 involves removing the class labelling and using numeric range outside 1 to 9 used in clean data (unadulterated) to set the threat threshold.

Fig.3 shows the preferred algorithm used in NETSQLIA with the gradient variations (0.78-0.914) in AUC overall performance between ANN (TCAP) = 0.912 and ML (TCLR) = 0.914 with 0.02 difference which tilts ANN towards 0.9 mark for a fair AUC against statistical ML. Table III details the calculation of evaluation results. The confusion matrix values *x* can be interpreted in percentages by a simple multiplication *(x*100)*.

Through repeated training of the supervised classifier using different attribute's items (rows) distribution, normalisation and split ratios, an optimum classification was achieved at: normalisation using transformation method of ZScore; split data ratio of 80:20 between training to test data. An optimised decent model of AUC = 0.78 has confusion matrix: accuracy = 0.911; precision = 0.967; recall = 0.937; and F1 Score = 0.952.

TABLE III. CONFUSION MATRIX

| Terminology | Formula | Values | Performance metrics | |
|---|---|---|---|---|
| True Positive (TP) | - | 9205 | *Accuracy (A)*= (TP+TN)/TE | 0.929 |
| True Negative (TN) | - | 84 | | |
| False Positive (FP) | - | 561 | | |
| False Negative (FN) | - | 151 | *Precision (P)*=TP/(PO) | 0.943 |
| Positive events (PE) | TP+FN | 9356 | | |
| Negative events (NE) | FP+TN | 645 | *Recall (R)*= TP/PE | 0.984 |
| + observations (PO) | TP+FP | 9766 | | |
| - observations (NO) | FN+TN | 235 | *F1Score=2* (R*P)/(R+P)* | 0.963 |
| Total events (TE) | PO+ NO | 10001 | | |

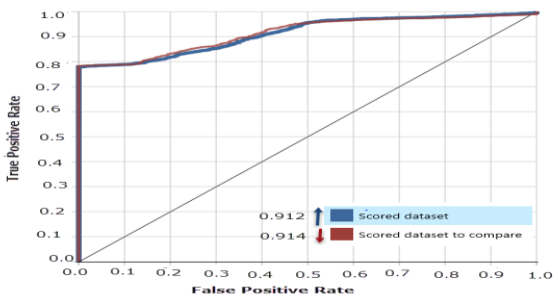


Fig.3 AUC comparison of TCAP(0.912) and TCLR(0.914) overall performance

## VI. CONCLUSION AND FUTURE WORK

We have demonstrated in NETSQLIA a model for numerical attributes extraction of any size from features primed as dataset items to ANN and cross validated in ML. The benchmark of detection rates against existing research works are not included here as this paper is aimed at determining the suitability of this novel attributes extraction technique to accurately train supervised learning model exposed as web services. The confusion matrix shown in Table III empirically evaluates the model.

The heavy-lifting done in NFA pattern matching reduces the classification to linear as against multi-class algorithm if NFA was not explored. NETSQLIA has built-in safeguards to group unrecognised patterns as unknown but future work is required to exhaustively map the random decimal values attributed to variations in attack features with exploring deep ANN and ML.

## References


[1] OWASP, "OWASP Top 10 - 2013," OWASP Top 10, p. 22, 2013.

[2] OWASP, "Web Application Firewall," 2015. [Online]. Available: https://www.owasp.org/index.php/Web_Application_Firewall. [Accessed: 15-Nov-2015].

[3] Microsoft Azure, "Microsoft Azure Machine Learning Studio," Microsoft Azure Machine Learning. [Online]. Available: https://studio.azureml.net/. [Accessed: 25-Jan-2015].

[4] Oracle, "SQL Injection Cookbook - Oracle - OWASP," OWASP, 2007. [Online]. Available: ttps://www.owasp.org/index.php/SQL_Injection_Cookbook_-_Oracle. [Accessed: 01-Jan-2015].

[5] Microsoft Azure, "Two-Class Averaged Perceptron," MSDN. [Online]. Available: https://msdn.microsoft.com/en-us/library/azure/dn906036.aspx. [Accessed: 19-Feb-2016].

[6] Microsoft Azure, "Two-Class Logistic Regression," MSDN Library,2015.[Online].Available:https://msdn.microsoft.com/en-us/library/azure/dn905994.aspx. [Accessed: 01-Feb-2016].

[7] W. G. J. Halfond, A. Orso, D. A. Kindy, and A. S. K. Pathan, "AMNESIA: Analysis and Monitoring for NEutralizing SQL-injection Attacks," Int. J. Commun. Networks Inf. Secur., vol. 5, pp. 80–92, 2013.

[8] S. J. Stolfo, "KDD cup 1999 dataset," UCI KDD Repos. http//kdd.ics.uci.edu, p. 0, 1999.

[9] L. K. Shar, L. C. Briand, H. K. Tan, and S. Member, "Hybrid Program Analysis and Machine Learning," vol. 12, no. 6, pp. 688–707, 2015.

[10] S. W. Boyd and A. D. Keromytis, "SQLrand: Preventing SQL injection attacks," in Applied Cryptography and Network Security, 2004, pp. 292–302.

[11] A. Liu, Y. Yuan, D. Wijesekera, and A. Stavrou, "SQLProb : A Proxy-based Architecture towards Preventing SQL Injection Attacks," System, pp. 2054–2061, 2009.

[12] C. Gould, Z. Su, and P. Devanbu, "JDBC checker: a static analysis tool for SQL/JDBC applications," in Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on, 2004, pp. 697–698.

[13] W. G. J. Halfond and A. Orso, "Preventing SQL Injection Attacks Using AMNESIA." Proc. 20th IEEE/ACM Int. Conf. Autom. Softw. Eng., pp. 174–183, 2005.

[14] MSDN, "Matching Behavior." [Online]. Available: https://msdn.microsoft.com/en-us/library/0yzc2yb0(v=vs.100).Aspx. [Accessed: 07-Jun-2015].

[15] E. Lawrence, "Fiddler free web debugging proxy," Telerik. [Online]. Available: http://www.telerik.com/fiddler. [Accessed: 11-Feb-2015].