

Assuring Privacy of AI-Powered Community Driven Android Code Vulnerability Detection

Janaka Senanayake¹[0000–0003–2278–8671],
Harsha Kalutarage¹[0000–0001–6430–9558],
Luca Piras²[0000–0002–7530–4119],
Mhd Omar Al-Kadri³[0000–0002–1146–1860], and
Andrei Petrovski¹[0000–0002–0987–2791]

¹ School of Computing, Robert Gordon University, Aberdeen AB10 7QB, UK
{j.senanayake,h.kalutarage,a.petrovski}@rgu.ac.uk

² Department of Computer Science, Middlesex University, London NW4 4BT, UK
l.piras@mdx.ac.uk

³ University of Doha for Science and Technology, Doha, Qatar
omar.alkadri@udst.edu.qa

Abstract. The challenge of training AI models is heightened by the limited availability of data, particularly when public datasets are insufficient. While obtaining data from private sources may seem like a viable solution, privacy concerns often prevent data sharing. Therefore, it is essential to establish a system that effectively balances privacy concerns with the need for data. In our previous work, we introduced “Defendroid”, which focuses on real-time Android code vulnerability detection using a blockchain federated neural network with explainable artificial intelligence. In this study, the Defendroid approach is enhanced by incorporating variable differential privacy techniques to ensure the privacy of the model training process. The proposed method significantly improves privacy, achieving a privacy budget between 1 and 1.5, while maintaining Defendroid’s baseline accuracy of 96% and an F1-Score of 0.96. As a result, this research thoroughly addresses concerns about the privacy of source code, filling a critical gap. This advancement not only showcases the effectiveness of the new approach but also its capability to address the significant challenges of privacy and data scarcity in AI-driven, community-focused Android code vulnerability detection.

Keywords: Android code vulnerability · federated learning · differential privacy · data scarcity · artificial intelligence

1 Introduction

The utilisation of AI has surged in recent years due to its rapid expansion and myriad features. Consequently, numerous fields are endeavoring to leverage AI to efficiently tackle various challenges. AI methodologies can also be employed to identify and promptly rectify source code vulnerabilities, ensuring the secure development of Android applications [22]. Despite several AI-based vulnerability

detection techniques being proposed, many of them fail to effectively detect code vulnerabilities during the coding phase. This limitation stems from the challenge of training such models with high accuracy, aggravated by the scarcity of available datasets, as many of these methods rely on classification-based approaches [7]. Even in cases where a properly labelled dataset is obtainable, it may be insufficient in size to significantly enhance the accuracy of the developed model. Therefore, addressing the issue of data scarcity is imperative.

Creating a sizable Android code vulnerability dataset or expanding an existing one by annotating source code from scanned Android Application Packages (APKs) is technically viable, but this method presents its own limitations. The breadth of the dataset, including the variety of distinct vulnerable categories, is constrained, and it might lack a sufficient number of code instances for novel vulnerabilities [23]. Another strategy involves training a model using source code directly sourced from app developers. However, developers might be hesitant to share their proprietary code due to privacy concerns. To address this challenge in the model training phase, federated learning offers a solution [27]. This technique involves dispersing the model training process across multiple entities interconnected within a federated environment. Consequently, these entities can individually train the model and refine the final model without divulging their data, which consists of source code snippets.

While federated learning offers a privacy-preserving training approach, it does not guarantee complete privacy, as attackers could potentially access model weights, identify training clients, and launch various attacks to uncover data [26]. Therefore, introducing noise into the model training process can significantly enhance privacy. To achieve this, differential privacy can be combined with federated learning for training the Android code vulnerability detection model which was proposed in our previous Defendroid approach [25]. Consequently, this paper introduces this approach to augment the existing model for Android code vulnerability detection, which was previously trained using the LVDAndro dataset [23] in our prior studies [12, 24].

In summary, this paper makes the following contributions:

- Extending our previous work, Defendroid [25], which introduced an AI-driven Android code vulnerability detection, to assure the model privacy.
- Simulating the application of differential privacy across various clients and analysing the variations in accuracy and F1-Score across the entire model.
- Integrating variable differential privacy to blockchain based federated averaging process by introducing DPV-BLFedAvg algorithm.

The paper is structured as follows: Section 2 provides the background and related work. Section 3 explains the methodology of how differential privacy applies to the existing Defendroid model. This section also presents results and corresponding discussions related to various experiments. The Section 4 covers conclusions and future work.

2 Background and Related Work

This section establishes the foundation for the study by discussing several key aspects, including the challenge of limited data availability in training AI models, the federated learning approach for AI model development, how blockchain can be used with federated learning, considerations of differential privacy, as well as AI-based Android code vulnerability detection alongside relevant literature.

2.1 Data Scarcity

Data scarcity in AI model development refers to the insufficiency of available data for effectively training machine learning algorithms. This shortage encompasses a lack of quantity, quality, and diversity in the datasets necessary to train models robustly [6]. Data scarcity poses significant challenges as it impedes the ability of AI models to generalise well to new, unseen data, often leading to over-fitting where models become overly tailored to the training data [3, 21]. Consequently, this limitation undermines the models' predictive accuracy and diminishes their applicability in real-world scenarios. Moreover, data scarcity restricts the complexity and capabilities of AI models, hindering their capacity to capture and represent intricate patterns in the data.

To address data scarcity, various strategies can be employed. Data augmentation techniques involve generating synthetic data or perturbing existing samples to enhance dataset size and diversity, thereby improving model robustness [4, 7]. Additionally, acquiring real-world data from diverse sources can help alleviate data scarcity by providing more comprehensive and representative datasets for model training [7]. However, the acquisition of real data may raise privacy concerns, highlighting the need to balance efforts to overcome data scarcity with considerations for ethical data practices. Thus, while addressing data scarcity is crucial for enhancing AI model performance, it must be done responsibly, taking into account privacy implications and ethical considerations.

2.2 Federated Learning for AI Models

Federated Learning is based on a distributed ML approach, which involves training multiple local models, on different devices, to create a global model. In a federated environment, clients who connect to the server can train their own local models with their own data in several training rounds. During these rounds, the model weights are shared with the federated server, which averages and updates them and creates a global model using the Federated Averaging (FedAvg) algorithm [28]. FedAvg is a popular FL averaging technique that facilitates local model training on multiple clients without sharing the client's local data with the server [15].

This approach offers the potential for model convergence with different client local data in non-independent and non-identically distributed settings. As a result, researchers from diverse fields have explored FL methods from various perspectives [13, 14]. However, none of these studies has examined how this can

be applied to AI-based Android code vulnerability detection models. Notably, federated learning only shares model weights with the federated server, not the original data, which effectively safeguards the privacy of the client source code in this case.

2.3 Blockchain-based Federated Learning

Conventional federated learning has certain inherent drawbacks, especially in the areas of coordinating participant activities, determining their benefits, and aggregating models. This is mainly because most current solutions depend on a centralised approach, which requires a reliable central authority for coordination. This approach brings with it several disadvantages, such as vulnerability to attacks, credibility issues, challenges in reward calculation, and difficulties in motivating participants for model training. Blockchain technology has been identified as a potential solution to these problems.

A blockchain is essentially a distributed ledger build with a series of data blocks arranged in sequence, each block containing a set of verifiable transactions [11]. Except for the first block, each block in a blockchain includes the hash value of the previous block’s header, forming a chain of linked blocks. Blockchains use unique consensus mechanisms like proof of work (PoW) or proof of stake (PoS), managed by a Peer-to-Peer (P2P) network of nodes. This structure makes it difficult to create but easy to verify each data block. The chain’s structure, the complexity of block creation, and the decentralised consensus achieved in P2P networks make the data in a blockchain highly resistant to changes. The unchangeable nature and traceability of data in blockchains make them a robust technical solution for maintaining a reliable database in a decentralised and trustworthy way [29]. As a result, researchers from various disciplines have examined methods of federated learning based on blockchain from different perspectives [13, 20]. However, except our previous work, Defendroid [25], none of the other studies have yet investigated the possible use of blockchain in conjunction with federated learning for detecting vulnerabilities in Android code using AI.

2.4 Differential Privacy

Differential privacy is a framework aimed at safeguarding individual data within a dataset while still enabling useful insights to be derived. By injecting calibrated noise into the training data, this approach ensures that even if an adversary gains access to the output of data analysis, they cannot discern whether specific individuals contributed to the dataset [26]. This protection reduces the risk of privacy breaches and unauthorised access to sensitive information. However, the incorporation of noise into the training process can impact the overall performance of AI models, necessitating a careful balance between privacy preservation and model utility [18].

Despite the utilisation of federated learning for AI model training, concerns persist regarding the privacy implications of transmitting model updates to a

central server. To address this issue, organisations can enhance privacy protection by incorporating differential privacy mechanisms into federated learning practices [19, 27]. This entails applying differential privacy techniques to the aggregation of model updates or parameters, ensuring the confidentiality of individual clients’ data throughout the federated learning process. By adopting this integrated approach, the risk of private information leakage is mitigated, thereby fortifying the overall privacy posture of federated learning models. This augmentation not only safeguards sensitive data but also instills greater confidence in stakeholders regarding the privacy-preserving capabilities of federated learning frameworks. As organisations navigate the evolving landscape of data privacy regulations and user expectations, the adoption of differential privacy in conjunction with federated learning represents a proactive step towards promoting responsible and ethical AI development practices.

2.5 Android Code Vulnerability Detection

Code vulnerabilities found within applications can lead to the emergence of bugs. These vulnerabilities encompass a spectrum of issues, ranging from simple programming errors like buffer overflows or inadequate input validation to more intricate concerns such as logic flaws or insecure cryptographic implementations [16]. Such vulnerabilities pose various security risks, including unauthorised access to sensitive data, denial of service attacks, and privilege escalation. The presence of code vulnerabilities in Android applications can be particularly concerning due to the prevalence of Android as the leading mobile operating system, with smartphones often serving as primary repositories of personal data [10, 22].

Developing tools for detecting Android code vulnerabilities using AI is a viable approach. To train such tools, a properly labelled dataset is required. Though there are some datasets such as Ghera and Androvul available, limitations such as labelling and data scarcity can be seen [22]. To overcome that the LVDAndroid dataset was introduced in our previous work [23], providing a labelled dataset that contains Android source code vulnerabilities. The LVDAndroid was produced by a combination approach of the MobSF [2] and Qark [17] scanners. The latest dataset was created by scanning the apps from repositories and labelled based on Common Weakness Enumeration (CWE). Since the LVDAndroid dataset is publicly available and provides good accuracy for vulnerability detection, it can be used as a valuable resource for training AI-based models. However, the concern of data scarcity is an issue to enhance the model and applicability of approaches such as federated learning and differential privacy need to be explored to overcome that.

3 Integrating Differential Privacy with Defendroid

The process of applying differential privacy was conducted to the model already proposed in Defendroid [25]. Initially, Defendroid establishes its backend model through the training of a neural network model using the LVDAndroid dataset.

This model is then trained within a blockchain federated environment. Subsequently, the best-performing model is seamlessly integrated into a Flask API. This API, in turn, is incorporated into Android Studio as a plugin, enhancing the developer's ability to detect vulnerabilities while leveraging the capabilities of XAI [25]. The Defendroid was able to achieve 96% accuracy with 0.96 F1-score for both binary classification to identify vulnerable codes, and multi-class classification to identify the CWE category of the vulnerability for 11 CWE classes. The overall process of the Defendroid is illustrated in Fig. 1. The federated learning simulation setup used in Defendroid is illustrated in Fig. 2.

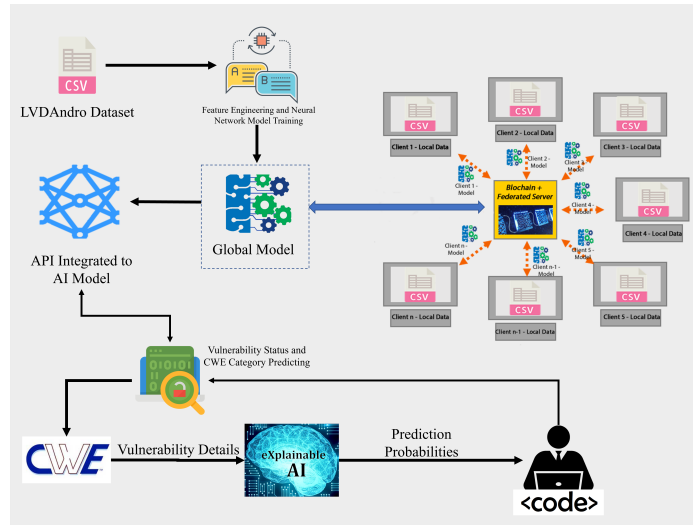


Fig. 1: Overview of the Defendroid Model [25]

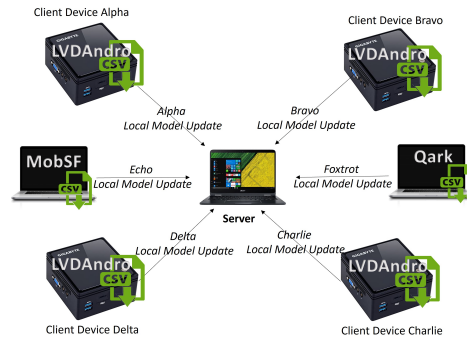


Fig. 2: Federated Learning Simulated Environment [25]

Though the prediction performance is high in the Defendroid model, privacy concerns and the confidentiality of the source code of the training clients was not addressed in detail. Hence, to enhance the confidentiality of the model training phase, techniques of differential privacy are introduced in this work. The integration of differential privacy allows for the quantification of privacy assurances. AI models can be responsibly crafted to learn from private data, thereby reducing the potential danger of revealing sensitive training data. After a model is trained using differential privacy, it remains unaffected by any individual training example or a small group of training examples in its dataset. This aids in reducing the risk of sensitive training data exposure in model training. This was achieved through the use of the Google differential privacy framework [1, 8] and the TensorFlow privacy framework [9].

The first step in this process was to execute a grid search to ascertain the most suitable values for parameters such as *learning rate*, *noise multiplier*, *l2_norm_clip*, and *num_microbatches* within differential privacy. The *l2_norm_clip* is a hyperparameter that sets the maximum Euclidean (L2) norm for each gradient applied to update model parameters, limiting the optimiser’s sensitivity to individual training points. The *noise_multiplier* refers to the quantity of noise sampled and added to gradients during training. Increased noise generally enhances privacy, potentially at the cost of lower utility. Data batches are divided into smaller units known as microbatches, each ideally containing a single training example. This allows gradients to be clipped on a per-example basis, reducing the negative impact of clipping on the gradient signal and typically maximising utility. The size of microbatches can be increased to include more than one training example to reduce computational overhead. The average gradient across these multiple training examples is then clipped. The *learning_rate*, a hyperparameter already present in neural network model, influences the impact of each update. A lower learning rate may aid convergence if the updates are noisy [9].

Under perfect conditions, not all participants in federated learning might have privacy as their main concern, with some potentially being more interested in contributing to open-source projects. As a result, it might not be essential to incorporate differential privacy for every client. To investigate this possibility, a series of experiments were conducted where the number of clients using differential privacy was varied. From the grid search the optimal values for this scenario were determined as *noise multiplier of 1.3*, *l2_norm_clip of 1.5*, *batch size of 256*, and *num_microbatches of 256*.

3.1 Applying Differential Privacy to Clients

Three experiments were conducted by applying differential privacy to various number of clients involved in the federated training process.

– Experiment 01:

As the first experiment, differential privacy was applied to all the six clients (a-Alpha, b-Bravo, c-Charlie, d-Delta, e-Echo, and f-Foxtrot) who involved with the initial Defendroid model’s Federated Learning environment [25]. As

a result, the models were named Defendroid-B-DP-a, Defendroid-B-DP-ab, Defendroid-B-DP-abcd, and so on, where “DP” signifies the use of differential privacy. This same experimental process was also used for the multi-class models, which were named Defendroid-M-DP-a, and so on. The results for the binary models can be found in Table. 1a, while the outcomes for the multi-class models are shown in Table. 1b.

Table 1: Experiment 01: Federated Learning with Differential Privacy Applied to All Clients

(a) Binary classification			(b) Multi-class classification		
Model Name	Accuracy	F1-Score	Model Name	Accuracy	F1-Score
Defendroid-B-DP-a	95.64%	0.9482	Defendroid-M-DP-a	93.01%	0.9201
Defendroid-B-DP-ab	95.81%	0.9492	Defendroid-M-DP-ab	93.17%	0.9212
Defendroid-B-DP-abc	95.86%	0.9496	Defendroid-M-DP-abc	93.33%	0.9242
Defendroid-B-DP-abcd	95.98%	0.9504	Defendroid-M-DP-abcd	93.45%	0.9315
Defendroid-B-DP-e	95.46%	0.9483	Defendroid-M-DP-e	92.11%	0.9156
Defendroid-B-DP-f	95.51%	0.9502	Defendroid-M-DP-f	92.31%	0.9167
Defendroid-B-DP-abcdef	95.65%	0.9501	Defendroid-M-DP-abcdef	94.51%	0.9411

– Experiment 02:

In the second experiment, differential privacy was applied to Alpha, Delta, Echo, and Foxtrot clients. This experiment was inspired by real-world situations where privacy concerns are a priority for some clients, but not for others. As a result, the models were named as Defendroid-B-DP-a, Defendroid-B-DP-a-N-b, Defendroid-B-DP-a-N-bc, Defendroid-B-DP-adeN-bc, and so on. In these labels, “DP” denotes the application of differential privacy, while “N” indicates clients that did not use differential privacy. The same experimental procedure was used for the multi-class models, which were labelled as Defendroid-M-DP-a and so on. The outcomes for binary classification are provided in Table. 2a, while the results for multi-class classification are outlined in Table. 2b.

Table 2: Experiment 02: Federated Learning with Differential Privacy Applied to Randomly Selected Clients

(a) Binary classification			(b) Multi-class classification		
Model Name	Accuracy	F1-Score	Model Name	Accuracy	F1-Score
Defendroid-B-DP-a	95.64%	0.9482	Defendroid-M-DP-a	93.01%	0.9201
Defendroid-B-DP-a-N-b	96.01%	0.9508	Defendroid-M-DP-a-N-b	93.4%	0.9261
Defendroid-B-DP-a-N-bc	96.05%	0.9523	Defendroid-M-DP-a-N-bc	94.00%	0.9359
Defendroid-B-DP-ad-N-bc	96.06%	0.9532	Defendroid-M-DP-ad-N-bc	94.20%	0.9410
Defendroid-B-DP-e	95.46%	0.9483	Defendroid-M-DP-e	92.11%	0.9156
Defendroid-B-DP-f	95.51%	0.9502	Defendroid-M-DP-f	92.31%	0.9167
Defendroid-B-DP-adeN-bc	95.81%	0.9541	Defendroid-M-DP-adeN-bc	94.78%	0.9452

– Experiment 03:

In the third experiment, differential privacy was applied to all clients with the exception of Alpha. The reason for this exclusion is that Alpha uses the entire LVDAndro dataset, which is publicly accessible. As a result, privacy considerations are not relevant for this client, while they are assumed for all other clients. In this context, the models were named as Defendroid-B-N-a, Defendroid-B-DP-b-N-a, Defendroid-B-DP-bc-N-a, Defendroid-B-DP-a-N-bcdef, and so on. Here also, “DP” stands for the application of differential privacy, and “N” denotes clients that did not use differential privacy. The same experimental procedure was used for the multi-class models, which were named Defendroid-M-DP-a and so on. The results for binary classification are provided in Table. 3a, while the findings for multi-class classification are outlined in Table. 3b.

Table 3: Experiment 03: Federated Learning with Differential Privacy Applied to All Clients Except Alpha

(a) Binary classification			(b) Multi-class classification		
Model Name	Accuracy	F1-Score	Model Name	Accuracy	F1-Score
Defendroid-B-N-a	96.04%	0.9574	Defendroid-M-N-a	93.50%	0.9213
Defendroid-B-DP-b-N-a	96.05%	0.9575	Defendroid-M-DP-b-N-a	94.1%	0.9281
Defendroid-B-DP-bc-N-a	96.06%	0.9577	Defendroid-M-DP-bc-N-a	94.5%	0.9397
Defendroid-B-DP-bcd-N-a	96.07%	0.9578	Defendroid-M-DP-bcd-N-a	94.67%	0.9410
Defendroid-B-DP-e	95.46%	0.9483	Defendroid-M-DP-e	92.11%	0.9156
Defendroid-B-DP-f	95.51%	0.9502	Defendroid-M-DP-f	92.31%	0.9167
Defendroid-B-DP-bcdef-N-a	96.08%	0.9580	Defendroid-M-DP-bcdef-N-a	95.33%	0.9502

The outcomes of these three experiments underscored the importance of balancing accuracy and privacy in model training since when applying differential privacy as above, the model performances have been comprised to some extent. The third experiment, which simulated real-world conditions by allowing client Alpha to use the LVDAndro dataset while other clients used their own data, demonstrated the efficacy of differential privacy measures in maintaining privacy.

The epsilon (ϵ) parameter, serving as a metric to gauge the privacy level of a model employing differential privacy known as privacy budget [8], was determined for the third experiment. The computed values were 1.19 for binary classification and 1.26 for multi-class classification. These findings indicate a strong privacy assurance, aligning with the optimal range of 1 to 1.5 for the privacy budget as suggested in [5].

3.2 Variation of Accuracy and Privacy Budget with Noise Multiplier

A study was carried out to determine the impact of noise multiplier variations on the performance of the model. This study entailed altering the noise multiplier within a range of 0 to 2.5 as this range is generally considered as optimal [5]. It compares the privacy budget (ϵ) and the accuracy for both binary and multi-class classification. Throughout this experiment, the `l2_norm_clip`, which is associated with gradient clipping for privacy preservation, was kept constant at 0. The outcomes of the binary classification are detailed in Table 4a and Table 4b details for the multi-class classification. Fig. 3 illustrates these variation.

Table 4: Variation of Accuracy and Privacy Budget with Noise Multiplier

(a) Binary Classification				(b) Multi-class Classification			
Noise Multiplier	Accuracy	Privacy Budget (ϵ)		Noise Multiplier	Accuracy	Privacy Budget (ϵ)	
0	96.17%	5.21		0	96.02%	6.35	
0.1	96.15%	5.01		0.1	96.01%	6.11	
0.3	96.13%	4.46		0.3	96.00%	5.51	
0.5	96.12%	3.23		0.5	95.98%	4.22	
0.7	96.11%	2.72		0.7	95.81%	3.55	
0.9	96.11%	2.11		0.9	95.77%	3.01	
1.1	96.10%	1.24		1.1	95.61%	2.22	
1.2	96.10%	1.2		1.2	95.44%	1.89	
1.3	96.08%	1.18		1.3	95.33%	1.22	
1.4	95.15%	1.13		1.4	94.42%	1.13	
1.5	95.11%	1.01		1.5	94.21%	1.09	
1.7	94.21%	0.93		1.7	94.02%	0.98	
1.9	93.67%	0.88		1.9	93.41%	0.91	
2.1	92.13%	0.85		2.1	92.97%	0.88	
2.3	91.55%	0.83		2.3	92.23%	0.84	
2.5	91.11%	0.81		2.5	91.09%	0.82	

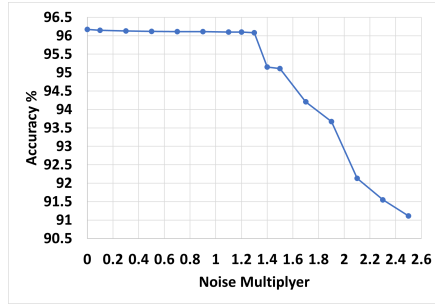
The findings indicated that a lower noise multiplier corresponds to higher accuracy but a lower privacy budget (indicated by a high ϵ value). Conversely, a high noise multiplier results in a high privacy budget (signified by a low ϵ value) but compromises accuracy. Thus, to ensure privacy without sacrificing too much accuracy, the noise multiplier should be adaptively adjusted to a balanced value.

3.3 Variation of Accuracy and Privacy Budget with L2_Norm_Clip

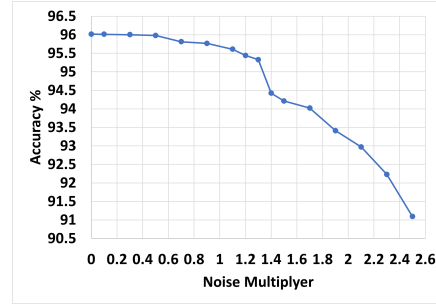
A similar experiment was carried out to examine the relationship between accuracy and the privacy budget (ϵ) with respect to `l2_norm_clip`. Throughout these experiments, the noise multiplier was maintained at 0.

The outcomes of the experiments of variation with `L2_Norm_Clip` for the binary classification are detailed in Table 5a, and Table. 5b details for the multi-class classification. Fig. 4 provides visual representations of these variation.

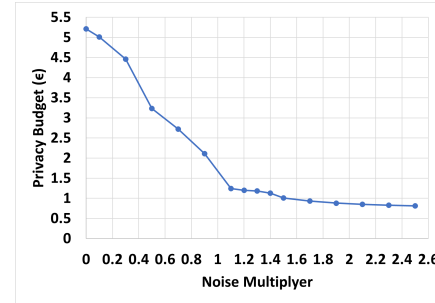
The analysis of the variations in accuracy and privacy budget, with respect to noise multiplier, and `l2_norm_clip`, reveals that a compromise in the performance



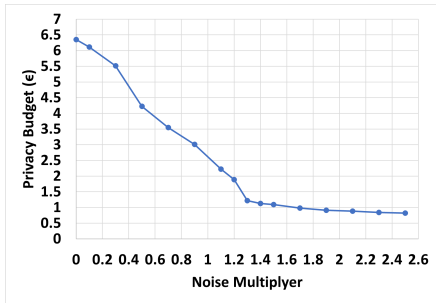
(a) Accuracy with Noise - Binary Classification



(b) Accuracy with Noise - Multi-class Classification



(c) Privacy Budget with Noise - Binary Classification



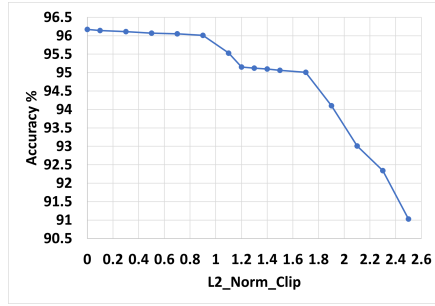
(d) Privacy Budget with Noise - Multi-class Classification

Fig. 3: Variation of Accuracy and Privacy Budget with Noise Multiplier

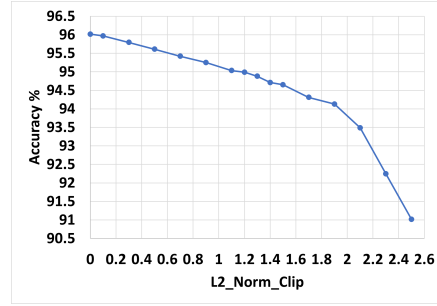
Table 5: Variation of Accuracy and Privacy Budget with L2_Norm_Clip

(a) Binary Classification			(b) Multi-class Classification		
L2_Norm_Clip	Accuracy	Privacy Budget (ε)	L2_Norm_Clip	Accuracy	Privacy Budget (ε)
0	96.17%	5.21	0	96.02%	6.35
0.1	96.14%	4.98	0.1	95.97%	6.01
0.3	96.11%	4.51	0.3	95.79%	5.42
0.5	96.07%	3.69	0.5	95.61%	5.03
0.7	96.05%	2.96	0.7	95.42%	4.21
0.9	96.01%	2.54	0.9	95.25%	3.76
1.1	95.53%	1.86	1.1	95.03%	3.06
1.2	95.15%	1.65	1.2	94.99%	2.11
1.3	95.12%	1.53	1.3	94.88%	1.49
1.4	95.10%	1.22	1.4	94.71%	1.21
1.5	95.06%	1.02	1.5	94.65%	1.02
1.7	95.01%	0.86	1.7	94.31%	0.83
1.9	94.10%	0.84	1.9	94.13%	0.74
2.1	93.01%	0.81	2.1	93.49%	0.62
2.3	92.34%	0.77	2.3	92.25%	0.56
2.5	91.03%	0.73	2.5	91.02%	0.48

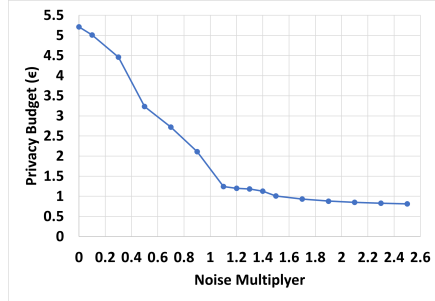
of the model is required to improve privacy. However, finding a balance between them is required by applying differently privacy in an adaptive manner.



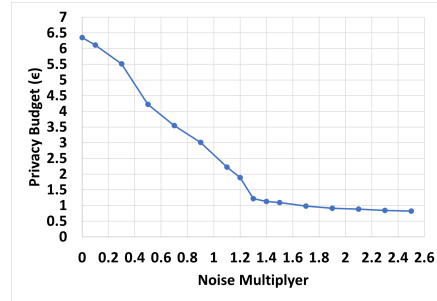
(a) Accuracy with L2_Norm_Clip - Binary Classification



(b) Accuracy with L2_Norm_Clip - Multi-class Classification



(c) Privacy Budget with L2_Norm_Clip - Binary Classification



(d) Privacy Budget with L2_Norm_Clip - Multi-class Classification

Fig. 4: Variation of Accuracy and Privacy Budget with L2_Norm_Clip

3.4 Differential Privacy Incorporating Variable Noise and L2_Norm_Clip with Blockchain-based Federated Learning

The ideal values for the noise multiplier and `l2_norm_clip` are contingent on the training data. Rather than maintaining these values as constants for all training datasets, they should be dynamically adjusted. However, this adjustment should be carried out in a way that does not compromise the performance of the models.

Therefore, a novel algorithm that takes all these factors into account was introduced. This algorithm is built upon the FedAvg algorithm and has the capability to dynamically adjust the noise multiplier and `l2_norm_clip` for clients who prioritise the privacy of their training data. With the use of this algorithm, it is possible to enhance or maintain the model accuracy and F1-Score while ensuring a high level of privacy, supported by the previously proposed blockchain consensus algorithm [25]. This newly proposed algorithm has been named the *Blockchain-based Federated Averaging with Differential Privacy incorporating Variable Noise and L2_Norm_Clip (DPV-BLFedAvg) Algorithm*, as referenced in Algorithm 1.

The DPV-BLFedAvg algorithm operates through a specified number of communication rounds in the federated learning setup and across all clients. For each

Algorithm 1: Blockchain-based Federated Averaging with Differential Privacy incorporating Variable Noise and L2_Norm_Clip (DPV-BLFedAvg) Algorithm

Input:

N : Total number of clients
 K : Number of communication rounds
 w_0 : Initial global model
 α : Learning rate
 MN : New Model
 MC : Current Model
 DP : Differential Privacy Status
 ϵ : Privacy Budget

Result: Updated global model w_K

Data:

$max_nmp = 2.5$: Max Noise Multiplier value
 $max_l2nc = 2.5$: Max L2_Norm_Clip value

```

1 loop1: for  $k = 1$  to  $K$  do
2   loop2: for  $i = 1$  to  $N$  do
3     if  $DP == TRUE$  then
4       loop3: for  $x = max\_nmp$  to  $x = 0$  decrement  $x$  by  $0.1$  do
5         loop4: for  $y = max\_l2nc$  to  $y = 0$  decrement  $y$  by  $0.1$  do
6            $opt = DPAdamOptimizer(x, y)$ 
7           Train a local model  $w_{i,k}$  using client  $i$ 's local data:
8            $w_{i,k} = LocalTraining(w_k, \alpha, opt)$ 
9           Aggregate local model updates:
10           $w_{agg} = \frac{1}{N} \sum_{i=1}^N w_{i,k}$  calculate  $\epsilon$ ;
11          Check the new model's Accuracy, F1-Score and Privacy:
12          if  $MN_{F1-Score} \geq MC_{F1-Score}$  and  $MN_{Accuracy} \geq MC_{Accuracy}$ 
             and  $1 \leq \epsilon \leq 1.5$  then
13            Add  $MN_{Weights}$  to the blockchain;
14             $MC \leftarrow MN$ ;
15            Update the global model:
16             $w_{k+1} = w_{agg}$ 
17            break iterations of loop3 and loop4;
18          else
19             $opt = AdamOptimizer()$ 
20            Train a local model  $w_{i,k}$  using client  $i$ 's local data:
21             $w_{i,k} = LocalTraining(w_k, \alpha, opt)$ 
22            Aggregate local model updates:
23             $w_{agg} = \frac{1}{N} \sum_{i=1}^N w_{i,k}$  If  $MN_{F1-Score} \geq MC_{F1-Score}$  and
               $MN_{Accuracy} \geq MC_{Accuracy}$  Add  $MN_{Weights}$  to the blockchain;
24             $MC \leftarrow MN$ ;
25            Update the global model:
26             $w_{k+1} = w_{agg}$ 

```

client, the algorithm first determines if the client has privacy concerns. If so, optimal values for the noise multiplier and `l2_norm_clip` are chosen. These values are automatically selected via nested loops, iterating from 2.5 to 0 for both the noise multiplier and `l2_norm_clip`, with each iteration round decreasing by 0.1. The DPAdamOptimiser is employed for model training. If the new model’s accuracy and F1-Score surpass those of the previous model and the privacy budget falls within the range of 1 to 1.5, the new model is updated in the blockchain, and the model weights are updated in the global model. If the client does not have privacy concerns, the accuracy and the F1-Score are the only factors compared after training the local model with AdamOptimiser.

This algorithm consistently preserves or enhances the existing level of accuracy and the F1-Score. If any client attempts to undermine the performance of the model, that model will be rejected and not incorporated into the global model. As a result, an accuracy of 96% and an F1-Score of 0.96 for both binary and multi-class classification are sustained, unless they can be improved further. Moreover, the privacy of the model is assured while achieving a privacy budget range of 1 to 1.5, which is deemed optimal.

To validate the algorithm, a series of experiments was carried out by incorporating six additional machines into the existing simulation architecture (Alpha, Bravo, Charlie, etc.). These new machines, designated as N1, N2, N3, N4, N5, and N6, each have a vulnerability dataset labelled using CWE-IDs. Some of these machines employ the LVDAndro approach for dataset generation, while others use source code that includes both vulnerable and non-vulnerable lines of code, labelled based on CWE IDs. The distributions of vulnerable and non-vulnerable code for these new clients can be found in Table 6.

Table 6: Statistics of New Clients’ Data

Characteristic	N1	N2	N3	N4	N5	N6
Vulnerable Code Lines	38,281	123,323	12,224	44,229	35,128	136,967
Non-Vul. Code Lines	97,124	445,235	66,245	125,941	66,324	236,932
Vul. Code Percentage	28.3%	21.7%	15.6%	26.0%	34.6%	36.7%
Distinct CWE-IDs	12	21	15	17	14	22

In these simulated experiments, the previous six clients (Alpha, Bravo, Charlie, etc.) did not have differential privacy applied. The new clients were tested under conditions both with and without differential privacy. In this context, the clients Alpha, Bravo, Charlie, Delta, Echo, and Foxtrot without differential privacy are referred to as *Old*. *DP-N1* refers to the N1 clients with differential privacy enabled, and *N-N2* refers to those without differential privacy. *B* denotes binary classification, and *M* represents multi-class classification. The results for binary classification can be found in Table 7, and those for multi-class classification are in Table 8.

Table 7: Evaluation of DPV-BLFedAvg Algorithm Results - Binary Classification

Model Name	Noise	L2_Norm_Clip	Accuracy	F1-Score	Privacy Budget (ϵ)	Model Status
Defendroid-B-Old	-	-	96.17%	0.9649	-	Accepted
Defendroid-B-Old-N-N1	-	-	96.17%	0.9650	-	Accepted
Defendroid-B-Old-DP-N1	0.5	0.4	96.17%	0.9649	1.2	Accepted
Defendroid-B-Old-N-N2	-	-	96.18%	0.9651	-	Accepted
Defendroid-B-Old-DP-N2	1.0	1.1	96.17%	0.9650	-	Accepted
Defendroid-B-Old-N-N3	-	-	96.17%	0.9623	-	Rejected
Defendroid-B-Old-DP-N3	0	0	96.17%	0.9623	5.6	Rejected
Defendroid-B-Old-N-N4	-	-	96.17%	0.9648	-	Rejected
Defendroid-B-Old-DP-N4	0	0	96.17%	0.9648	4.3	Rejected
Defendroid-B-Old-N-N5	-	-	96.17%	0.9649	-	Accepted
Defendroid-B-Old-DP-N5	0	0	96.17%	0.9649	3.8	Rejected
Defendroid-B-Old-N-N6	-	-	96.18%	0.9652	-	Accepted
Defendroid-B-Old-DP-N6	1.3	1.5	96.17%	0.9650	1.3	Accepted

Table 8: Evaluation of DPV-BLFedAvg Algorithm Results - Multi-class Classification

Model Name	Noise	L2_Norm_Clip	Accuracy	F1-Score	Privacy Budget (ϵ)	Model Status
Defendroid-M-Old	-	-	96.02%	0.9624	-	Accepted
Defendroid-M-Old-N-N1	-	-	95.99%	0.9601	-	Rejected
Defendroid-M-Old-DP-N1	1.5	2.3	95.51%	0.9588	1.4	Rejected
Defendroid-M-Old-N-N2	-	-	96.03%	0.9631	-	Accepted
Defendroid-M-Old-DP-N2	1.9	1.8	96.03%	0.9625	1.3	Accepted
Defendroid-M-Old-N-N3	-	-	95.88%	0.9593	-	Rejected
Defendroid-M-Old-DP-N3	0	0	95.88%	0.9593	3.1	Rejected
Defendroid-M-Old-N-N4	-	-	96.02%	0.9625	-	Accepted
Defendroid-M-Old-DP-N4	0	0	96.02%	0.9625	2.2	Rejected
Defendroid-M-Old-N-N5	-	-	95.79%	0.9391	-	Rejected
Defendroid-M-Old-DP-N5	0	0	95.79%	0.9391	1.7	Rejected
Defendroid-M-Old-N-N6	-	-	96.03%	0.9626	-	Accepted
Defendroid-M-Old-DP-N6	1.1	1.3	96.03%	0.9624	1.4	Accepted

The experiments revealed that the base performance of the model consistently maintained a minimum accuracy of 96.17%, an F1-Score of 0.9640 for binary classification, and an accuracy of 96.02%, with an F1-Score of 0.9624 for multi-class classification. When differential privacy was applied, an additional check was conducted to measure the privacy budget, which should also fall within the range of 1 to 1.5. If either of these conditions including minimum accuracy, minimum F1-Score, or privacy budget range was not met, the new model would be rejected and the global model would not be updated. If all the conditions are satisfied, then the global model will be updated with the new model and weights are shared across the all clients.

Utilising differential privacy will lengthen the model training process due to the algorithm’s need for extra iterations to pinpoint the optimal combination of noise and the `L2_norm_clip` necessary to meet the desired privacy budget. Nonetheless, this extension in training time will not affect the prediction speed for identifying Android code vulnerabilities. Consequently, developers will observe no efficiency discrepancy upon integrating this into Defendroid’s backend and can seamlessly continue with the Android code vulnerability detection process as outlined in [25].

4 Conclusion and Future Work

Developing AI-based models, including the specific research problem tackled in this work, faces a significant challenge stemming from the limited availability of training data due to privacy concerns surrounding data sharing. In our previous work, we addressed this challenge by incorporating federated learning into a blockchain-based environment, known as the Defendroid, to encourage more clients to participate in model training while safeguarding their privacy. To enhance the privacy of the Defendroid model further, this study introduces an adaptive implementation of differential privacy within the blockchain-based federated learning setup. This ensures the protection of individual data contributors’ privacy, even when sharing a substantial amount of aggregate data. This approach significantly advanced the research process, resulting in a model that achieved a high accuracy of 96% and an F1-Score of 0.96 as baseline prediction values for both binary and multi-class classification, while maintaining a reasonable privacy budget. In conclusion, this study presents a robust AI-based model that preserves privacy while identifying vulnerabilities in the source code of Android applications. Not only does this privacy-focused model surpass its predecessor, but it also maintains a low privacy budget, addressing concerns surrounding data privacy. The success of this approach highlights its potential to transform AI model training in contexts where privacy and data scarcity are critical concerns. Future enhancements could involve fine-tuning training parameters and expanding the community driven federated learning environment to a broader audience.

References

1. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. p. 308–318. CCS '16, Association for Computing Machinery, New York, NY, USA (2016). <https://doi.org/10.1145/2976749.2978318>, <https://doi.org/10.1145/2976749.2978318>
2. Abraham, A., Magaoferi, Dobrushin, M., Nadal, V.: Mobile security framework (mobsf) (2023), <https://github.com/MobSF/Mobile-Security-Framework-MobSF>, accessed: 2023-03-11
3. Alzubaidi, L., Bai, J., Al-Sabaawi, A., Santamaría, J., Albahri, A., Al-dabbagh, B.S.N., Fadhel, M.A., Manoufali, M., Zhang, J., Al-Timemy, A.H., et al.: A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications. *Journal of Big Data* **10**(1), 46 (2023). <https://doi.org/10.1186/s40537-023-00727-2>
4. Anaby-Tavor, A., Carmeli, B., Goldbraich, E., Kantor, A., Kour, G., Shlomov, S., Tepper, N., Zwerdling, N.: Do not have enough data? deep learning to the rescue! In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 7383–7390 (2020)
5. Andrew, G., Thakkar, O., McMahan, B., Ramaswamy, S.: Differentially private learning with adaptive clipping. *Advances in Neural Information Processing Systems* **34**, 17455–17466 (2021)
6. Babbar, R., Schölkopf, B.: Data scarcity, robustness and extreme multi-label classification. *Machine Learning* **108**(8-9), 1329–1351 (2019). <https://doi.org/10.1007/s10994-019-05791-5>
7. Bansal, M.A., Sharma, D.R., Kathuria, D.M.: A systematic review on data scarcity problem in deep learning: Solution and applications. *ACM Comput. Surv.* **54**(10s) (sep 2022). <https://doi.org/10.1145/3502287>, <https://doi.org/10.1145/3502287>
8. Google: Differential privacy (2023), <https://github.com/google/differential-privacy>
9. Google: Tensorflow privacy (2024), <https://github.com/tensorflow/privacy>
10. Gupta, A., Suri, B., Kumar, V., Jain, P.: Extracting rules for vulnerabilities detection with static metrics using machine learning. *International Journal of System Assurance Engineering and Management* **12**(1), 65–76 (2021). <https://doi.org/10.1007/s13198-020-01036-0>
11. Jiang, S., Cao, J., Wu, H., Yang, Y., Ma, M., He, J.: Blochie: A blockchain-based platform for healthcare information exchange. In: 2018 IEEE International Conference on Smart Computing (SMARTCOMP). pp. 49–56 (2018). <https://doi.org/10.1109/SMARTCOMP.2018.00073>
12. Kalutarage, H., Senanayake, J., Al-Kadri, M.O., Petrovski, A., Piras, L.: Android code vulnerabilities early detection using ai-powered acved plugin. In: Atluri, V., Ferrara, A.L. (eds.) *Data and Applications Security and Privacy XXXVII*. pp. 339–357. Springer Nature Switzerland, Cham (2023)
13. Li, L., Fan, Y., Tse, M., Lin, K.Y.: A review of applications in federated learning. *Computers & Industrial Engineering* **149**, 106854 (2020). <https://doi.org/10.1016/j.cie.2020.106854>
14. Li, Q., Wen, Z., Wu, Z., Hu, S., Wang, N., Li, Y., Liu, X., He, B.: A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering* (2021)

15. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine* **37**(3), 50–60 (2020). <https://doi.org/10.1109/MSP.2020.2975749>
16. Lin, G., Wen, S., Han, Q.L., Zhang, J., Xiang, Y.: Software vulnerability detection using deep neural networks: A survey. *Proceedings of the IEEE* **108**(10), 1825–1848 (2020). <https://doi.org/10.1109/JPROC.2020.2993293>
17. LinkedIn: Quick android review kit (qark) (2015), <https://github.com/linkedin/qark/>, accessed: 2023-04-02
18. Liu, B., Ding, M., Shaham, S., Rahayu, W., Farokhi, F., Lin, Z.: When machine learning meets privacy: A survey and outlook. *ACM Computing Surveys (CSUR)* **54**(2), 1–36 (2021)
19. Mothukuri, V., Parizi, R.M., Pouriyeh, S., Huang, Y., Dehghantanha, A., Srivastava, G.: A survey on security and privacy of federated learning. *Future Generation Computer Systems* **115**, 619–640 (2021)
20. Qu, Y., Uddin, M.P., Gan, C., Xiang, Y., Gao, L., Yearwood, J.: Blockchain-enabled federated learning: A survey. *ACM Comput. Surv.* **55**(4) (nov 2022). <https://doi.org/10.1145/3524104>, <https://doi.org/10.1145/3524104>
21. Sambasivan, N., Kapania, S., Highfill, H., Akrong, D., Paritosh, P., Aroyo, L.M.: “everyone wants to do the model work, not the data work”: Data cascades in high-stakes ai. In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI ’21, Association for Computing Machinery, New York, NY, USA (2021). <https://doi.org/10.1145/3411764.3445518>, <https://doi.org/10.1145/3411764.3445518>
22. Senanayake, J., Kalutarage, H., Al-Kadri, M.O., Petrovski, A., Piras, L.: Android source code vulnerability detection: A systematic literature review. *ACM Comput. Surv.* **55**(9) (jan 2023). <https://doi.org/10.1145/3556974>, <https://doi.org/10.1145/3556974>
23. Senanayake, J., Kalutarage, H., Al-Kadri, M.O., Piras, L., Petrovski, A.: Labelled vulnerability dataset on android source code (lvdandro) to develop ai-based code vulnerability detection models. In: *Proceedings of the 20th International Conference on Security and Cryptography - SECRIPT*. pp. 659–666. INSTICC, SciTePress (2023). <https://doi.org/10.5220/0012060400003555>
24. Senanayake, J., Kalutarage, H., Petrovski, A., Al-Kadri, M.O., Piras, L.: Fedrevan: Real-time detection of vulnerable android source code through federated neural network with xai. In: *Computer Security. ESORICS 2023 International Workshops*. pp. 426–441. Springer Nature Switzerland, Cham (2024)
25. Senanayake, J., Kalutarage, H., Petrovski, A., Piras, L., Al-Kadri, M.O.: Defendroid: Real-time android code vulnerability detection via blockchain federated neural network with xai. *Journal of Information Security and Applications* **82**, 103741 (2024). <https://doi.org/https://doi.org/10.1016/j.jisa.2024.103741>, <https://www.sciencedirect.com/science/article/pii/S2214212624000449>
26. Wang, Y., Wang, Q., Zhao, L., Wang, C.: Differential privacy in deep learning: Privacy and beyond. *Future Generation Computer Systems* **148**, 408–424 (2023). <https://doi.org/https://doi.org/10.1016/j.future.2023.06.010>, <https://www.sciencedirect.com/science/article/pii/S0167739X23002315>
27. Wei, K., Li, J., Ding, M., Ma, C., Yang, H.H., Farokhi, F., Jin, S., Quek, T.Q.S., Vincent Poor, H.: Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security* **15**, 3454–3469 (2020). <https://doi.org/10.1109/TIFS.2020.2988575>

28. Zakariyya, I., Kalutarage, H., Al-Kadri, M.O.: Resource efficient federated deep learning for iot security monitoring. In: Li, W., Furnell, S., Meng, W. (eds.) *Attacks and Defenses for the Internet-of-Things*. pp. 122–142. Springer Nature Switzerland, Cham (2022). https://doi.org/10.1007/978-3-031-21311-3_6
29. Zhu, J., Cao, J., Saxena, D., Jiang, S., Ferradi, H.: Blockchain-empowered federated learning: Challenges, solutions, and future directions. *ACM Comput. Surv.* **55**(11) (feb 2023). <https://doi.org/10.1145/3570953>, <https://doi.org/10.1145/3570953>