2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) | 978-1-6654-1658-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/TRUSTCOM53373.2021.00097

AndroCreme: Unseen Android Malware Detection Based on Inductive Conformal Learning

Gang Zhang*†§, Hao Li *†§, Zhenxiang Chen *‡¶, Lizhi Peng *†, Yuhui Zhu *†, Chuan Zhao *†,
*Shandong Provincial Key Laboratory of Network Based Intelligent Computing, University of Jinan, China
†School of Information Science and Engineering, University of Jinan, China
‡school of cyber science and technology, Shandong University, China
¶Corresponding author, Email: czx@ujn.edu.cn
§Equal contribution

***Abstract*—Android platform is facing serious malware threats due to its popularity, as evidenced by the drastic increase on the number of mobile malware families and variants in recent years. Detecting malware variants and zero-day malware is a critical challenge that must be addressed to protect mobile devices against malware attacks. In this study, we present AndroCreme, a novel network intrusion detection system (NIDS) that can identify unseen malware by analyzing the network behavior of Android malware. To address the temporal bias issue in NIDS, we propose a method for rapid iterative update of the model based on data selection and data size limitation. The selection of effective data is carried out by induction and conformal technology, and the data scale is controlled by the method of time window and data cycle selection. To further achieve fast training speed and high efficiency, we leverage a gradient boosting framework that uses a tree-based learning algorithm, namely, LightGBM, as the meta predictor. We evaluate the performance of AndroCreme over 400K real-world network flows, which are collected from over 30K Android benignware and 21K malware applications. The experimental results show that, compared with the retraining method using all data, AndroCreme requires only a small amount of datareduce more than 3x to obtain better detection performance, which effectively solves the temporal bias.**

***Index Terms*—Android, NIDS, unseen malware detection, model updating, Time window, data selection**

I. Introduction

Mobile security is one of the critical issues in security sphere. It was reported that one in 36 devices used in organizations is under a high risk [1], and the number of attacks using malicious mobile software is nearly doubled in just one year [2]. Monet [3] shows that 97% of mobile malware originates from the Android platform, and 99% of the mobile malware was found in third-party app stores since these app stores usually don't have a sufficient malware screening mechanism.

To counter the aforementioned threats, on one hand, several Android malware detection approaches [4–6] have been proposed based on machine learning techniques to analyze static or dynamic features of malware. However, to prevent malware from being detected by detection technology, attackers constantly change the malware's attack behavior[7, 8] (code obfuscation, repackage the applications, etc.) and also add various evasion strategies (e.g., detect software execution scenarios). It can be found that popular attacks are not constant.The malware shows great changes, that is, new families appear, spread, and disappear as time flows, and may have shorter life cycle. This trait of malware causes the detection performance of malware predictors based on learning and content information to continue to decline or even fail over time. Therefore, to retain high performance, the detection models have to be upgraded and retrained frequently. Meanwhile, another promising approach for Android malware detection focuses on analyzing malicious network behavior of the malware [9–13]. By learning the pattern of the malicious network behavior, NIDS can identify novelty malicious traces among benign network samples. Coincidentally, NIDS confronts the above same issue with time elapsed, which can make the model invalid as well.

With the time elapsed, unseen samples of network behaviors can have a different distribution from the training set of existing model, such as the appearance of new samples or the change of labels of existing samples, which in turn leads to a decrease in model detection performance, we call this problem is temporal bias[14, 15]. To detect Android malware with temporal bias, Work [16] proposed Hoeffding Anytime Tree, a novel incremental decision tree learning algorithm ,which has natural adaptability to temporal bias. Bifet A et al [17] proposed an adaptive RandomForest algorithm, which uses the ADWIN [18] extension to improve the RandomForest algorithm, While ensuring the detection effect of the model, the incremental update of the model is completed. In this paper, we propose a novel malware detection system by dissecting network traffic behavior, AndroCreme, which through data selection and data scale control to realize the rapid update of the model, thereby solving the problem of temporal bias. Different from [9, 13], AndroCreme uses network statistical features instead of content-based features to protect user's privacy. Moreover, we adopt a tree-based learning algorithm, specifically, LightGBM [19], which is a highly efficient gradient boosting decision tree, as the meta classifier to perform classification.

We conduct comprehensive experiments to evaluate AndroCreme's effectiveness and robustness in detecting malicious unseen samples. The experiments use over 400K (70K malicious and 330K benign) network flows which are collected from over 30K benignware and 21K malware of Android. In addition, the datasets we used involve a medley of older and newer Android samples, between 2012 and August 2018, as

2324-9013/21/$31.00 ©2021 IEEE
DOI 10.1109/TrustCom53373.2021.00097

the meta data to assess performance of the AndroCreme in confront of temporal bias.

Summary of Contributions – The major contributions of this paper are as follows:

- In this paper, we propose AndroCreme, a network intrusion detection system which is able to solve temporal issues. The framework uses conformal prediction technology to associate each prediction with a statistically valid confidence level to quickly identify new effective data and retrain the model to maintain the predictor's detection performance.
- In order to realize the rapid iterative update of the model, we analyzed both new and old data from the time axis and proposed an effective data loop selection method based on time window, which can limit the data size and maintain the detection performance of the updated model.
- We introduce a novel and largest volume Android network traces dataset, on which we conduct comprehensive experiments to verify the effectiveness of our proposed method. Compared with the model update using the entire training set, AndroCreme can get an improvement 1% to 2% in various indicators when the data set is reduced by nearly 3x.

Organization – The remainder of this paper is organized as follows. In Section II, we briefly review the temporal bias challenge and present an overview of the AndroCreme framework. The details of the dataset to be used for experiments were described in Section III. In Section IV, we present the details of the evaluation experiments and results. Discussions and limitations are shown in Section VI. Section V gives the related work and we conclude our work in Section VII.

II. The AndroCreme System

A. Overview

Temporal bias is a common phenomenon which conforms with the fact of realistic scenarios. In this paper, to tackle the issue above-mentioned, we propose AndroCreme, which is a novel malicious network traffic detection system by virtue of grasp the difference between malicious and benign behavior. Figure 1 illustrates the whole model update process of the proposed system. Depicted the diagram, in step (1), we capture the network traffic of the Android application by deploying DroidCollector[20] and split it into network flows.In step (2), we extract and select statistical features from each network flow. In addition, the time-window of the dataset is also determined at this step. Then in step (3), we leverage gradient boosting tree-based learning classification algorithm, LightGBM, to build a classifier in our system and detect malware. In step (4), we use the classifier trained in step (3) to classify unseen samples into 2 classes – benign or malicious. In step (5), we use inductive conformal technique and data loop selection method based on time-window to assess and rebuild our predictor to retain the detection performance with temporal bias.

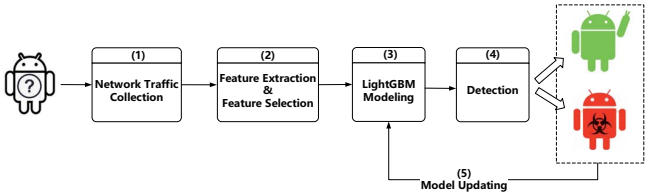


Fig. 1. Overview of AndroCreme System.

B. Network Traffic Collection

Network traffic from malwares comprises the basic training samples in our system. We use DroidCollector [20], a high-performance Android network traffic collection framework, which deployed in our laboratory to automatically collect network traces. We use DroidCollector to collect packet-level flows generated during the first 5 minutes from both malware and benignware samples, and store traffic data as a `pcap` file.

C. Features Extraction & Features Selection

After the procedure of network traffic collection, gathered network traffic data through a traffic collection platform produced several mixed network session flows data, which are generated during the first 5 minutes of each application operation. AndroCreme first splits TCP flows from a `pcap` file by virtue of five-tuple (source/destination IP address, source/destination port, and protocol), and then uses `joy`[1] developed by Cisco to extract original features from each single flow. Then we analyze and extract features from network flow data. We extract over 79 network statistical features without content-based information by implementing self-defined parameters, which prevents leakage of users' privacy. In addition, we further select features based on important weights by employing a meta-transformer. Details of the selected features description shows in Section IV-C.

D. Model training and detection based on LightGBM

AndroCreme constructs a detection model by leveraging LightGBM [19] technique, which is a tree-based learning algorithm by applying gradient boosting and leaf-wise tree growth algorithm. LightGBM is proposed to tackle the problem of efficiency and scalability in high feature dimensions with intensive computation consumption in the process of tree point splitting. LightGBM leverages Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) methods to compute information gain with a very low computation complex. GOSS is a balanced algorithm to reduce the amount of data and ensure accuracy. EFB can bind many mutually exclusive features into one feature, so as to achieve the purpose of dimensionality reduction. In addition, LightGBM is an ensemble learning method composed of many sub-classifiers, which provides the possibility of parallel training of the model. Hence, we use LightGBM as a classifier to reduce training time consumption and improve model performance.

[1]https://github.com/cisco/joy

E. Model Updating

1) Inductive Conformal: Machine learning-based predictor is promising in handle network traffic classification challenges, but there are no recurrence events or samples in case of online malicious traces detection. With the time elapsed, the predictor's detection performance continues to decrease or even fail due to the temporal bias problem. In order to solve the issue aforementioned, AndroCreme uses an online learning technique based on inductive conformal [21]. Inductive conformal[2] can associate the model's prediction for each sample with the statistically valid confidence. Given a test sample x_i and a user-specified significance level $\epsilon \in (0,1)$, inductive conformal outputs a prediction region $\Gamma_i^\epsilon \subseteq Y$ that contains the true output value $y_i \in Y$ with probability $1-\epsilon$. Therefore, AndroCreme uses the idea of inductive conformal, by calculating p-value, to select data with new knowledge for the next model update. Here the p-value calculation we used is as follows.

$$error(y_i|x_j) = 0.5 - \frac{\hat{P}(y_i|x_j) \ - \ max_{y\,!=\,y_i}\hat{P}(y|x_i)}{2}, \tag{1}$$

$$H\,(x|h) = \begin{cases} 0 & ,x < h \\ 1 & ,x > h \end{cases}, \tag{2}$$

$$P(y_i|x_j) = \frac{\sum_0^n (H(error(y_i|x_{k!=j})|error(y_i|x_j)))}{n}, k = 0,1,2...,n. \tag{3}$$

where $error(y_i|x_j)$ is the margin error, i is the label number, j is the sample number, $\hat{P}(y_i|x_j)$ is the probability that sample x_j is predicted to be label y_i. For example, in the binary classification, $\hat{P}(y_0|x_0)$ represents the probability of y_0 predicted by sample 0, and $\hat{P}(y_1|x_0)$ represents the probability of y_1 predicted by sample 0. $H\,(x|h)$ is the 0-1 function, when $x < h$, take 0, otherwise take 1, and n is the number of samples in the test set. Through the above formula, we can calculate the statistical p value of each sample, that is, $P(y_i|x_j)$. For example, in the binary classification, $P(y_0|x_0 s)$ represents the possibility that sample 0 belongs to y_0, and $P(y_1|x_0)$ represents the possibility that sample 0 belongs to y_1.

By setting a user-specified significance level $\epsilon \in (0,1)$, we can calculate whether a sample x belongs to a certain label in the $1-\epsilon$ confidence prediction region. For example, for the binary classification problem, exists sample x, if $P(y_0|x)$ falls in the rejection region and $P(y_1|x)$ falls in the acceptance region, we denote it as $[false, true]$. Then we believe that those samples predicted to output as $[false, fasle]$ neither conform to the distribution of existing positive data, nor the distribution of existing negative data, but carry new knowledge not available in existing data. The samples whose prediction is $[true, false]$ or $[false, true]$ are in accordance with one of the existing positive and negative data distributions, these samples carry existing knowledge that can be distinguished by existing data, here we discard them. The samples predicted as $[true, true]$ fit both the existing positive data distribution and the existing negative data distribution, these samples carry complex knowledge that is difficult to distinguish by model trained with existing data. By setting the significance level and calculating the p-value, we can quickly screen out those samples with new and complex knowledge(The main reason for temporal bias), re-add those samples to the training, to deal with the impact of temporal-bias on the model prediction.

In addition, the inductive conformal technology selects data based on the model prediction probability. LightGBM is a tree-based integrated model, and the sample prediction probability is based on the training data set, which can minimize the influence of the model itself on data selection. However, if the test results of the model trained based on the LightGBM algorithm are poor, we suggest to find new features or choose other models.

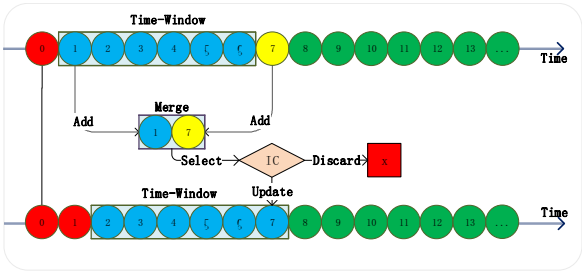


Fig. 2. Training set update based on time window and data loop selection method.(IC:Inductive Conformal)

2) Data loop selection based on time-window: Although the inductive conformal only selects data with new knowledge and complex knowledge, if the selected data is simply added to the training set, the size of the training set will become larger and larger as time goes by. In response to above problems, we design a method of data loop selection using time-window to limit the size of the training data set while retaining historical knowledge and removing data that is different from the current data distribution. As shown in Figure 2, each circle represents a time slice of data (or extracted features), the yellow circle represents the data that has just been predicted, the green circle represents the data that has not been predicted, the red circle represents the discarded data, and the blue circle represents the data in the time-window. Before the time window moves forward, the time slice data at the end of the time window is merged with the predicted time-sliced data. Here, merge the data in the blue circle 1 and the yellow circle 7 in the figure, and then select the merged data through inductive conformal (IC) to obtain data with new and complex knowledge, the blue circle 7 in the figure. At the same time, discard the repeated data, the red x square in the figure.

By setting a fixed time window, the issue of unlimited growth of the training set is solved. By merging the test data with the data corresponding to the end time of the time window, and select the merged data through inductive conformal, we can retain historical valid data as well as reduce

[2]https://github.com/donlnz/nonconformist/

the negative impact of old data on model performance to a certain extent. In addition, in a specific environment, the data distribution over a period of time is relatively stable, which determines the convergence of the model that has learned all the knowledge after several updates.

III. Datasets and Examples

In this section, we depict the dataset we employed to assess AndroCreme system's performance in face of temporal bias. We leverage over 67,000 Android applications between 2012 and August 2018 which gathered from Drebin [22], VirusShare[3], AndroZoo [23], and AnZhi market[4].

Benignware and Malware. We deploy VirusTotal analysis API [24] to label Android applications we have. Different from Tesseract [15], in order to ensure labels more persuasive, we set over 8 positives value as the threshold of malware, positives value less than 2 (zero or one) as the threshold of benignware, and positives values between 2 and 8 as the threshold of greyware. In the end, we get over 30,000 benignware and 21,000 malware as the meta data set in our evaluation experiments, and over 24% (about 16K) of the Android applications are labelled as greyware which are not containing in our evaluation experiments section.

Network Traffic Data. We leverage DroidCollector in AndroCreme system to collect network traces of the malware and benignware which labeled by above-mentioned technique. DroidCollector use multiple virtual machine to collect traffic, and to collect effective network traffic, we adopt real Android device rather than virtual machine, namely, NOKIA X6, to collect real-world traffic data. Over 1,450,000 TCP flows are collected leveraging our traffic collection system. Furthermore, VirusTotal APIs are also implemented in our traffic labelling module. Through label engineering of flows, we have got over 400,000 labeled network flows (malicious and benign) with different time periods. We collect over 68,000 malicious flows (which contains about 4,000 and 64,000 flows with 443 and 80 destination port) and 330,000 benign flows (which contains about 88,000 and 242,000 flows with 443 and 80 destination port) generated by the malware and benignware, respectively.

TABLE I
Datasets Overview

Time	Apps		Flows	
	benignware	malware	benign	malicious
2012	2798	2325	9311	3185
2013	4654	3116	21426	11755
2014	8037	4087	65342	8671
2015	5855	3986	49681	5949
2016	4473	2676	48398	3957
2017	2224	1592	36725	2318
2018	2009	3518	72784	32772

[3]https://virusshare.com/
[4]http://www.anzhi.com/

Table I shows an overview of the dataset of benignware and malware within different time periods and per year distribution of benign/malicious traffic in our dataset.

Temporal Bias Setting: Experimental network traces data set contains samples between 2012 and August 2018, to verify robustness of the AndroCreme, we leverage network samples before 2018 as the training data, and 2018's later for testing with 8 different evaluation experiments.

A. Examples

From the data description aforementioned, we dig out some interesting findings and in this section we will illustrate them. Figure 3 depicts the tendency of the malicious host between 2012 and 2017 overlaps with 2018, and we notice that malicious host We obtain 38 malicious hosts extracted from 2012 are reduplicating with 2018, comparing with 90 in 2017.

In addition, Figure 4 shows 2 examples of malicious `host` corresponding different malware families in 2018 and contrasts with before. Figure 4(a) and 4(b) depicts *m.aedxdrcb.com* correlates with many malware families before 2018 more than during 2018, and the typical malicious behavior transformed from family *Gen* to *smsreg*, which means that a number of malware families can change the malicious behavior which never sees before with time elapsed. Figure 4(c) and 4(d) describes *pay.91mgame.com* corresponding families are increasing between 2018 and before, which means that the malicious behavior become more complicate.

The findings above-mentioned illustrates that temporal bias plays an significant role in hampering model's detection performance, hence, both researchers and industrial practitioners should consider and improve the predictor's performance in realistic scenarios.

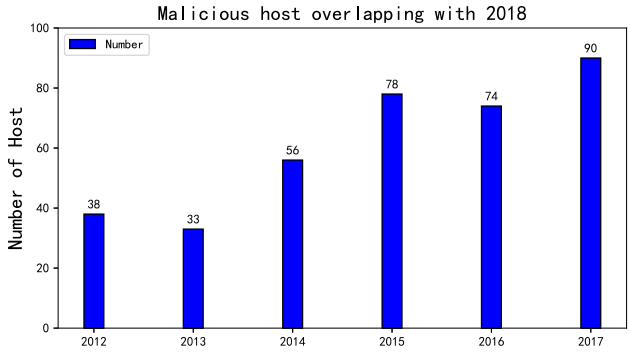
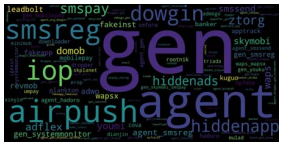


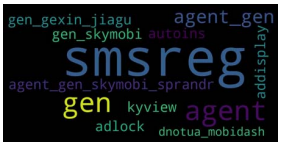
Fig. 3. Tendency of malicious host overlaps with 2018.

IV. Evaluation

In this section, we evaluate the effectiveness and robustness of the AndroCreme framework in terms of malicious traces detection. First, we introduce different evaluation metrics used in this paper. Next we measure AndroCreme's performance in online malicious traces detection mode and evaluate effective

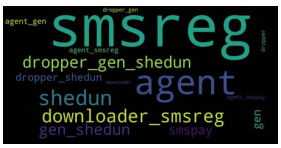


(a) Host: *m.aedxdrcb.com* different families between 2012 and 2017



(b) Host: *m.aedxdrcb.com* different families in 2018

(c) Host: *pay.91mgame.com* different families between 2012 and 2017



(d) Host: *pay.91mgame.com* different families in 2018

Fig. 4. Different families distribution with different Host.

of model's updating technique. Then we describe the impact of the feature selection and analyze how network statistical traffic features works in proposed method. Furthermore, We also evaluate performance of the AndroCreme system in face of temporal bias challenge.

All experiments are conducted on a quad-core 3.20 GHz PC with 8 GB RAM, Ubuntu 16.04 operating system. The proposed system implementation is using Python language. In particular, network traffic preparation module we use tshark[5] with Lua[6] dissector, overall of the system's code above 1,500 lines.

A. Performance Evaluation

To measure the performance of the proposed AndroCreme system, we use the following metrics: Accuracy, Precision, Recall, F-measure. The accuracy is the overview evaluation metric of the model in the prediction results both malicious and benign, Precision can indicate the proportion of actual malware predicted to be malware, and the recall can indicate the proportion of the actual number of malicious which are correctly identified. In addition, F-measure is a harmonic mean between precision and recall and we assign the same weights to both precision and recall in our evaluation performance. Different performance metrics above-mentioned are defined as follows.

$$Accuracy = \frac{TP\ +\ TN}{TP +\ TN +\ FP +\ FN}, \tag{4}$$

$$Precision = \frac{TP}{TP +\ FP}, \tag{5}$$

$$Recall = \frac{TP}{TP +\ FN}, \tag{6}$$

$$F - measure = \frac{2(Precision\ *\ Recall)}{Precision\ +\ Recall}, \tag{7}$$

[5]https://www.wireshark.org/docs/man-pages/tshark.html
[6]https://www.lua.org/

where the true positive (TP) is the number of actual malicious classified as malicious, true negative (TN) is the number of actual benign samples classified as benign ones, false positive (FP) is the number of actual benign samples classified as malicious, and false negative (FN) is the number of actual malicious classified as benign samples.

In addition, we also use Area Under Time (AUT) metric, the area under the performance curve over time, which is a time-aware performance metric with considering time decay and proposed by [15]. AUT metric is defined as follows.

$$AUT(f, N) = \frac{1}{N-1} \sum_{k=1}^{N-1} \frac{[f(x_{k+1}) + f(x_k)] * (1/N)}{2}. \tag{8}$$

where $f(x_k)$ is the meta quantitative measure performance (in this paper, we use F-measure as the f), and N is the number of time slots. AUT range 0 to 1, and the ideal predictor with robustness to time decay in the time spanning is $AUT = 1$.

B. Significance levels ϵ setting

Before the comprehensive experiment, we discuss, when model updates, the amount of newly added data and the corresponding detection performance under different significance levels ϵ. We use the data before 2018 as the training set and the monthly data from January to August 2018 as the test set. The performance metric is the mean value of the monthly detection performance metric. The experimental results are shown in Figure II, where NU represents no model update, FU represents model update using all data, and ICU represents the model update under different significance levels ϵ using the inductive conformal technology. The comparison of the experimental results of ICU, NU, and FU show the effectiveness of using inductive conformal technology to select data, that is, only a small amount of data is required to greatly improve the detection performance of the model, and this improvement is very close to the performance obtained by training with all data. Comparing the ICU experimental results under different significance levels $\epsilon = 0.04, 0.05...$, as the number of new data increases, the model detection performance increases slowly, indicating that only a small part of the data has a decisive effect on the improvement of the model performance. In addition, through the comparison of the experiment results of NU($\epsilon = 0.2$) and ICU($\epsilon = 0.2$), indicating that the model update can continuously improve the knowledge contained in itself, thereby reducing the demand for new data. The experimental results show that the value of ϵ gradually stabilizes after being greater than 0.1. We recommend that the setting range of ϵ is 0.05 to 0.25. If you want more stable performance, you can increase the value of ϵ. In the following comparison of experiments, We set the significance level $\epsilon = 0.2$ to improve the fault tolerance of the model towards the data set.

C. Feature Selection and Time-window

Table IV shows the number of features we selected and the time window size of the determined data set. Among them, Selector represents the feature selector, Number represents

TABLE II
MODEL TRAINING DATA COST AND PERFORMANCE COMPARISON UNDER DIFFERENT SIGNIFICANCE LEVELS ϵ

Method	performance		Data	
	Accuracy	F-measure	Add-Number	Add-Ratio
FU	0.9572	0.9124	10110	1.0
NU($\epsilon = 0.20$)	0.8882	0.8024	5324	0.4877
ICU($\epsilon = 0.04$)	0.9517	0.9014	1024	0.1036
ICU($\epsilon = 0.05$)	0.9532	0.9063	1213	0.1231
ICU($\epsilon = 0.06$)	0.9519	0.9022	1321	0.1362
ICU($\epsilon = 0.08$)	0.9548	0.9082	1629	0.1697
ICU($\epsilon = 0.10$)	0.9559	0.9100	1829	0.1922
ICU($\epsilon = 0.15$)	0.9556	0.9122	2377	0.2493
ICU($\epsilon = 0.20$)	0.9558	0.9111	2905	0.3072
ICU($\epsilon = 0.30$)	0.9573	0.9140	3930	0.4175

the number of features, and Time-Window represents the time period of the training set. All experiments use LightGBM as the classifier. Training set, test set and performance metrics are consistent with section IV-B. Test 1 shows the effects of features selected by different feature selectors and different dimensional features selected by the same selector on model performance. Here, because LightGBM is an ensemble model based on decision tree, we use the number of times each feature is used when the decision tree is split to evaluate the importance of the feature. The result of feature selection are shown in Table III. Test 2 shows the performance trend of the model under the training set of different time-windows(the model not update). The experimental results show that a longer time window can not achieve better results and the data that far away do not always have a positive impact on the prediction effect of the model. For the setting of the time window, we suggest that if the data is prone to concept drift with time, reduce the time window, otherwise increase the time window.

D. Evaluation of AndroCreme robustness in temporal bias and model update training cost

In this section, we evaluate the effectiveness and robustness of the AndroCreme malware detection system in confront of temporal bias. In our evaluation experiment, we use the real-world network dataset and combining the conclusions drawn from the experimental results in section IV-C, we use a 5-year time window, with the data from 2013-2018 as the training set, the data from January to August 2018 as the test set, the mean of the test set results as our measure. As shown in Table V, NU, FU and ICU are as described in the section IV-B, and WINU represents model update using time window to limit the data set. AndroCreme is our method. Section II-E introduces the specific update process of AndroCreme. Add is the number of samples added by the model update.

The experimental results of NU and FU proved the existence of the temporal bias and show the importance of adding new data to retrain the model. The experimental results of FU and ICU prove the effectiveness of using inductive conformal to select data with new knowledge and complex knowledge. When the evaluation metrics of ICU and FU are basically the same, the data volume of ICU is only 0.3472 of FU. The

TABLE III
STATISTICAL FEATURES DESCRIPTION

Index	Statistical Features	Weight
1	std of the byte distribution	0.0710
2	time to live of ip packet from $da \rightarrow sa$	0.0664
3	skewness of the byte distribution	0.0576
4	mean of the byte distribution	0.0504
5	std of the data field byte number array in each TCP packet	0.0476
6	sum of the upload bytes in UDP Data field	0.0468
7	entropy in bits per byte	0.0453
8	first window size of the download tcp packet	0.0396
9	total entropy which over all of the bytes in the flow	0.0391
10	sum of the download bytes in UDP Data field	0.0335
11	bytes sent from $da \rightarrow sa$	0.0315
12	window scale of the upload tcp packet	0.0303
13	mean of the download data field inter-packet time	0.0294
14	std of the byte array distribution	0.0289
15	max of the byte distribution	0.0287
16	byte distribution coefficient of variation	0.0284
17	std of the download data field inter-packet time	0.0281
18	the download data field inter-packet time	0.0278
19	mean of the number of bytes in data field	0.0278
20	entropy of the byte distribution	0.0262
21	bytes sent from $sa \rightarrow da$	0.0252
22	maximum of the bytewise Walsh-Hadamard Transform of the Data fields	0.0252
23	seconds to live of the ip packet	0.0248
24	byte distribution variance of the k-statistic	0.0248
25	circular standard deviation of the byte distribution	0.0242
26	circular mean of the byte distribution	0.0217
27	maximum segment size of the upload tcp packet	0.0200
28	number of packets from $da \rightarrow sa$	0.0168
29	sum of the byte distribution	0.0165
30	time since start of flow	0.0163

TABLE IV
FEATURE SELECTION AND TIME-WINDOW DETERMINATION.(RF:RANDOMFOREST,LGM:LIGHTGBM)

Test-one			Test-two	
F-measure	Selector	Number	F-measure	Time-Window
0.7633	none	79	0.7783	2011 - 2018
0.7699	rf	39	0.7801	2012 - 2018
0.7700	adaBoost	21	**0.7896**	**2013 - 2018**
0.7715	**lgm**	**30**	0.7800	2014 - 2018
0.7652	lgm	28	0.7457	2015 - 2018
0.7648	lgm	32	0.7153	2016 - 2018
0.7590	lgm	38	0.6567	2017 - 2018

experimental results of FU and WINU prove the limited effect of old data on model updating, and the evaluation results of the model have not changed significantly. The comparison of the experimental results of FU and AndroCreme shows that larger training not always make the classifier better, the size of the time window, and the selection of valid data are also important factors that need to be considered.

E. Evaluation of AndroCreme in the stability of detection effect

In this section, we will analyze the stability of our method from the smaller granular test results. Figure 5 shows the monthly data test results of NU, FU and AC on precision and F-measure indicators. As shown in Figure 5, AC has an outstanding effect in maintaining model stability, exceeding the theoretical optimal FU update. The experimental results further illustrate that the old data will have a negative impact

TABLE V
COMPARISON OF THE EFFECTS OF DIFFERENT UPDATE METHOD.(AC:ANDROCREME)

Accuracy	Recall	F-measure	Add	Add-Rate	method
0.8840	0.6963	0.7896	-	-	NU
0.9654	0.9090	0.9325	10283	1.0	FU
0.9656	0.9099	0.9318	10283	1.0	WINU
0.9644	0.9076	0.9321	3468	0.3472	ICU
0.9738	**0.9220**	**0.9418**	**3617**	**0.3061**	**AC**

on the model, and AC can reduce the impact to a certain extent through data selection. In addition, we quantified this predicted trend using the AUT[15] metric, and we took a more representative metric (F-measure) to calculate the AUT area. Through calculation, the AUT area of NU, FU and AC methods are **0.782**, **0.928** and **0.939**, respectively.

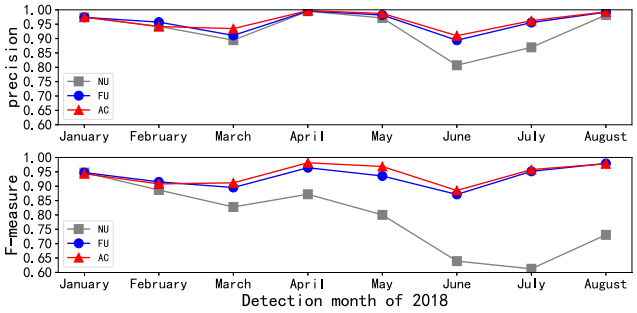


Fig. 5. AndroCreme detects monthly performance trends and stability comparison.(AC:AndroCreme)

Figure 6 shows the variety in the number of data selected by AC in the model update. Where, Add represents the number of newly added data for each model update, Sum represents all the data that has not been selected for each model update, and Add-Rate represents the proportion of selected data to the total number of people. It can be seen that AC can greatly reduce the number of data required for model update, and through Add-Rate, it can be seen that the proportion of growth data is slowly decreasing, which shows that the model is continuously converging in the update.

F. Comparison of AndroCreme with other methods

Here we choose EFDT [16] and ARF [17] two methods to compare with, in order to reduce the contrast interference factor, here we use all 79 features. Figure 7 shows the performance results of the model update of different methods. It is obvious that AndroCreme achieves better detection results than the simple model adaptive update strategy. EFDT is a tree structure model, because the data imbalance and the double influence of the structure of the model itself lead to unacceptable results. ARF is an integrated learning method, and the model itself has good generalization. It can be seen that the same adaptive method ARF surpasses EFDT in all aspects, and ARF has also achieved a certain improvement

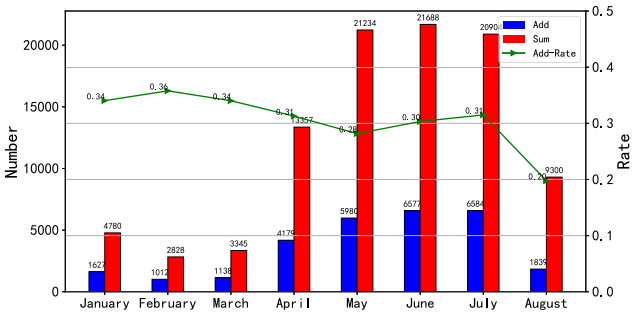


Fig. 6. Model update data growth trend of AndroCreme method.(the significance level $\epsilon = 0.2$)

compared to NU. However, ARF has not gotten rid of the negative impact of data imbalance on model performance, and the difference between Precision and Recall is huge. We found that AndroCreme can reduce the impact of unbalanced data sets on the model. In the process of data selection by AndroCreme, a large number of duplicate samples will be removed, which balances the data set to a certain extent. In addition, for the experimental data set, from a purely model point of view, the overall performance of lightGBM is better than RF.

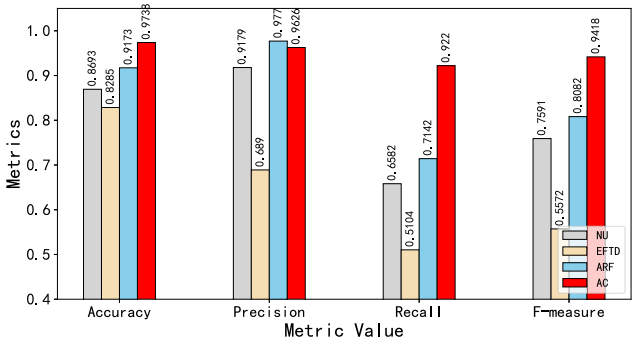


Fig. 7. Comparison of AndroCreme with other methods.(the significance level $\epsilon = 0.2$)

V. LIMITATIONS

The experimental evaluation results demonstrate the effectiveness of AndroCreme in network intrusion detection with temporal bias, however, the proposed framework still have several limitations.

Starvation in labeled data. We collect an enormous number of Android application dataset, however, the dataset contains many ambiguous and vague applications (which we called greyware) can not used in evaluation experiments due to different anti-virous scanners may has various detection reports. Furthermore, because of AndroCreme's label engineering module is based on `Host` scanning by VT, only TCP flows with `443` and `80` port are using in evaluation experiments.

Hence, a lot of valuable network traces will not consider in this paper and may hamper system's identification performance. In addition, AndroCreme's model update is a fixed time which is artificially set, and unnecessary update operations will inevitably lead to a waste of resources.

To augment size of the training dataset and improve AndroCreme detection performance, in our future work, we are going to study UDP-based flows and label these data set as the training data into model training module, and we will combine data distribution detection technology to determine when the model is updated. In addition, We also need to research zero-day malware which the Anti-virous scanners with uncertain detection result.

VI. Conclusion

This paper proposed a novel network intrusion detection framework, AndroCreme, to detect Android malware with temporal bias issues. For purpose of the goal above-mentioned, we use inductive conformal technology to select effective data carrying new and complex knowledge and use a data loop selection method based on time windows to suppress the growth of the data set size. AndroCreme adopts over 400,000 real-world network flows data sets which generate from over 30K benignware and 21K malware of Android applications. AndroCreme has achieved outstanding results by considering design in three aspects(feature, model, and data). In experiment evaluation, Compared with the non-update model(NU), AndroCreme only needs to add a small amount of new data, which improved 3.31% and 15.22% in precision and F-measure, respectively. Even compared with the theoretically optimal model trained with all data(FU), the indicators of our method have improved by 1%-2%. In the stability evaluation, AC obtained the largest AUT area: 0.939.

VII. Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grants No.61672262 and No. 61702218, Project of Shandong Province Higher Educational Youth Innovation Science and Technology Program under Grant No.2019KJN028, Project of Independent Cultivated Innovation Team of Jinan City under Grant No.2018GXRC002,Shandong Provincial Key Research and Development Project under Grant No. 2019GGX101028, Shandong Province Higher Educational Science and Technology Program under Grant No. J18KA349, Natural Science Foundation of Shandong Province under Grant No. ZR2019LZH015.

References

[1] symantec. https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf.

[2] kaspersky. The number of mobile malware attacks doubles in 2018, as cybercriminals sharpen their distribution strategies.

[3] Mingshen Sun, Xiaolei Li, John C. S. Lui, Richard T. B. Ma, and Zhenkai Liang. Monet: A user-oriented behavior-based malware variants detection system for android. *IEEE Transactions on Information Forensics* & Security, PP(99):1–1, 2016.

[4] Parvez Faruki, Ammar Bharmal, Vijay Laxmi, Vijay Ganmoor, Manoj Singh Gaur, Mauro Conti, and Muttukrishnan Rajarajan. Android security: A survey of issues, malware penetration, and defenses. *IEEE Communications Surveys & Tutorials*, 17(2):998–1022, 2017.

[5] Lucky Onwuzurike, Enrico Mariconti, Panagiotis Andriotis, Emiliano De Cristofaro, Gordon J. Ross, and Gianluca Stringhini. Mamadroid: Detecting android malware by building markov chains of behavioral models (extended version). *ACM Trans. Priv. Secur.*, 22(2):14:1–14:34, 2019.

[6] Talha Ongun, Timothy Sakharaov, Simona Boboila, Alina Oprea, and Tina Eliassirad. On designing machine learning models for malicious network traffic classification. *arXiv: Cryptography and Security*, 2019.

[7] Vaibhav Rastogi, Yan Chen, and Xuxian Jiang. Catch me if you can: Evaluating android anti-malware against transformation attacks. *IEEE Transactions on Information Forensics* & Security, 9(1):99–108, 2013.

[8] Xiao Chen, Chaoran Li, Derui Wang, Sheng Wen, Jun Zhang, Surya Nepal, Yang Xiang, and Kui Ren. Android hiv: A study of repackaging malware for evading machine-learning detection. *IEEE Transactions on Information Forensics and Security*, 15:987–1001, 2020.

[9] Shanshan Wang, Zhenxiang Chen, Qiben Yan, Bo Yang, Lizhi Peng, and Zhongtian Jia. A mobile malware detection method using behavior features in network traffic. *Journal of Network and Computer Applications*, 133:15 – 25, 2019.

[10] Michal Piskozub, Riccardo Spolaor, and Ivan Martinovic. Malalert: Detecting malware in large-scale network traffic using statistical features. *ACM SIGMETRICS Perform. Eval. Rev.*, 2019.

[11] Shanshan Wang, Qiben Yan, Zhenxiang Chen, Lin Wang, Spolaor Riccardo, Bo Yang, and Mauro Conti. Lexical mining of malicious urls for classifying android malware. In *SecureComm, 2018 Proceedings*, 2018.

[12] Bushra A AlAhmadi and Ivan Martinovic. Malclassifier: Malware family classification using network flow sequence behaviour. In *Proc. of IEEE eCrime*, 2018.

[13] Shanshan Wang, Zhenxiang Chen, Qiben Yan, Ke Ji, Lizhi Peng, Bo Yang, and Mauro Conti. Deep and broad url feature mining for android malware detection. *Information Sciences*, 513:600–613, 2020.

[14] W.-B Pan, G Cheng, X.-J Guo, and S.-X Huang. An adaptive classification approach based on information entropy for network traffic in presence of concept drift. *Jisuanji Xuebao/Chinese Journal of Computers*, 40:1556–1571, 2017.

[15] Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. Tesseract: Eliminating experimental bias in malware classification across space and time. In *28th USENIX Security Symposium*, 2019.

[16] Albert Bifet, Jiajin Zhang, Wei Fan, Cheng He, Jianfeng Zhang, Jianfeng Qian, Geoff Holmes, and Bernhard Pfahringer. Extremely fast decision tree mining for evolving data streams. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1733–1742, 2017.

[17] Heitor M Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrcio Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9-10):1469–1495, 2017.

[18] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM, 2007.

[19] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc., 2017.

[20] Dong Cao, Shanshan Wang, Qun Li, Zhenxiang Chen, Qiben Yan, Lizhi Peng, and Bo Yang. Droidcollector: A high performance framework for high quality android traffic collection. In *2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, August 23-26, 2016*, pages 1753–1758, 2016.

[21] *Conformal prediction*, pages 17–51. Springer US, Boston, MA, 2005.

[22] Daniel Arp, Michael Spreitzenbarth, Malte Hbner, Hugo Gascon, and Konrad Rieck. Drebin: Effective and explainable detection of android malware in your pocket. In *Network and Distributed System Security Symposium*, 2014.

[23] Kevin Allix, Tegawend F. Bissyand, Jacques Klein, and Yves Le Traon. Androzoo: collecting millions of android apps for the research community. In *Mining Software Repositories*, 2017.

[24] Virustotal. https://www.virustotal.com/.