applied sciences

MDPI

Article

Cybersecurity Threats Based on Machine Learning-Based Offensive Technique for Password Authentication

Kyungroul Lee [1] and Kangbin Yim [2,*]

[1] R&BD Center for Security and Safety Industries (SSI), Soonchunhyang University, Asan-si 31538, Korea; carpedm@sch.ac.kr
[2] Department of Information Security Engineering, Soonchunhyang University, Asan-si 31538, Korea
* Correspondence: yim@sch.ac.kr; Tel.: +82-41-530-1741

Received: 10 January 2020; Accepted: 11 February 2020; Published: 14 February 2020

Abstract: Due to the emergence of online society, a representative user authentication method that is password authentication has been a key topic. However, in this authentication method, various attack techniques have emerged to steal passwords input from the keyboard, hence, the keyboard data does not ensure security. To detect and prevent such an attack, a keyboard data protection technique using random keyboard data generation has been presented. This technique protects keyboard data by generating dummy keyboard data while the attacker obtains the keyboard data. In this study, we demonstrate the feasibility of keyboard data exposure under the keyboard data protection technique. To prove the proposed attack technique, we gathered all the dummy keyboard data generated by the defense tool, and the real keyboard data input by the user, and evaluated the cybersecurity threat of keyboard data based on the machine learning-based offensive technique. We verified that an adversary obtains the keyboard data with 96.2% accuracy even if the attack technique that makes it impossible to attack keyboard data exposure is used. Namely, the proposed method in this study obviously differentiates the keyboard data input by the user from dummy keyboard data. Therefore, the contributions of this paper are that we derived and verified a new security threat and a new vulnerability of password authentication. Furthermore, a new cybersecurity threat derived from this study will have advantages over the security assessment of password authentication and all types of authentication technology and application services input from the keyboard.

Keywords: vulnerability analysis; password authentication; machine learning; user authentication

1. Introduction

Due to the emergence of online society, a representative user authentication method that is the password authentication method has been presented [1]. This method registers a password by the user, and then authenticates the user by comparing the registered password with the input password. Therefore, the information that must be protected in this authentication method is the input password. This password is generally input from the keyboard, therefore a technique is required to protect the data input through the keyboard.

Various attack techniques have appeared in the past, and the key-logger is a representative attack tool [2]. The tool records all keyboard data input by the user, and is easily available from the Internet. Moreover, new attack techniques have been introduced by attackers, such as WinProc replacement, keyboard message hooking, filter driver insertion, interrupt object replacement, interrupt descriptor table (IDT) replacement, direct polling, and C/D (control/data) bit vulnerability exploitation techniques.

The WinProc replacement attack technique steals keyboard data by replacing a window procedure, whereas the keyboard message hooking attack technique steals keyboard data by hooking a keyboard event message. These two techniques are attack techniques in user mode. If the defender applies the technique to protect the keyboard data in the same user mode, the attacker and the defender compete in the same mode, which causes the attacker to fail the keyboard data exposure attack.

An attacker overcomes the defense technique by attacking in kernel mode with higher privilege than user mode [3]. The filter driver insertion attack technique steals keyboard data by inserting a filter driver in the PS/2 keyboard device driver stack. The interrupt object replacement attack technique replaces an object handling an interrupt associated with a PS/2 keyboard, and steals keyboard data. The IDT replacement attack technique steals keyboard data by replacing the table handling interrupt associated with the PS/2 keyboard. These three techniques attack in kernel mode. Therefore, when the defender also applies methods to prevent the exposure of the keyboard data in the same kernel mode, the attacker and the defender have a race condition, which causes the attacker to fail the keyboard data exposure attack.

An attacker overcomes the defense technique by carrying out a hardware access-based attack. The direct polling attack technique exploits keyboard data by periodically accessing the input and output memory associated with the PS/2 keyboard [4]. This attack preempts keyboard data by monitoring keyboard input before the data input from the keyboard arrives at the kernel mode interrupt handling routine, thereby defeating the defense technique in kernel mode [5].

To counteract such hardware access-based attacks, a keyboard data protection technique that utilizes random keyboard data generation has been developed to prevent attackers from stealing the exact input keyboard data by the user [6]. The key concept of this technique prevents exposure of the real keyboard data input by the user, but rather it detects the attack techniques of keyboard data. Specifically, the defender invokes a keyboard input event forcibly by generating random keyboard data that protects the actual keyboard data input by the user by filtering the generated keyboard data. The defender knows the random keyboard data it generates, and can differentiate the actual data from the random data. At the same time, the attacker uses a direct polling attack technique to collect both the dummy data and the keyboard data input by the user. However, the attacker can hardly differentiate the two.

To prevent the failure of the direct polling attack, an attack technique, known as the C/D bit vulnerability exploitation technique, using a feature that appears when generating random keyboard data has emerged [7]. This technique takes advantage of the vulnerability of the C/D bit in the keyboard controller status register to steal keyboard data, and neutralizes the keyboard protection technique that utilizes random keyboard data generation. Nevertheless, this attack technique causes a lot of overload on the system due to the condition of periodically setting the C/D bit, and preemptively collecting all the input keyboard data. Consequently, from the attacker's point of view, there is a need for an attack technique that classifies random keyboard data, and steals keyboard data input from the user, without generating an overload on the system.

In this study, we propose a method for classifying random keyboard data and keyboard data input by a user to analyze the vulnerability of keyboard data. We demonstrate the feasibility of classifying keyboard data using machine-learning models based on attack techniques, such as filter driver insertion, interrupt object replacement, and direct polling, to collect keyboard data. For vulnerability analysis, the available data features are the elapsed time of keyboard data acquisition, collected keyboard data and flags (parity error, receive time-out, transmit time-out, inhibit switch, C/D, system flag, input buffer full (IBF), and output buffer full (OBF)). Among these features, utilizing the C/D bit provides a complete classification of the actual keyboard data input by the user. However, as described above, overload on the system occurs, due to various conditions, and there is a risk that an attack is detected by judging according to abnormal behavior and access. Therefore, we demonstrate the feasibility of classifying the keyboard data input by the user based on machine-learning models, focused on the entire keyboard data collected.

The contributions of this paper are as follows:

- We proposed an attack method to classify random keyboard data based on machine learning using existing attack techniques that do not succeed in the keyboard data exposure attack, and by using machine learning, verify that the attacker can steal the keyboard data even if the attack technique that makes it impossible to attack keyboard data exposure is used.
- In this study, we focused on how the defender generates random keyboard data, and determined that the defender calls the keyboard data generation function periodically, such as a timer. Moreover, we analyzed the data availability of flags collected from the keyboard controller. Therefore, a dataset is constructed by collecting the elapsed time of keyboard data acquisition, collected keyboard data, and flags as data. As a result, this proves the capacity to effectively classify random keyboard data by using the dataset proposed in this study.
- In this paper, we derived and verified the security threat and vulnerability of the password authentication method. The proposed method is very effective with an accuracy of 96.2%. Specifically, it is possible to steal the user's password. Furthermore, this attack method means that there is a security threat and vulnerability in the password authentication method. Conclusively, a new cybersecurity threat derived from this study will have advantages over the security assessment of the password authentication method.

The rest of the paper is organized as follows. Section 2 describes the keyboard data transmission process and conventional keyboard data attack and defense techniques. Section 3 introduces the configured attack system keyboard data dataset. The experimental results of the keyboard data attack using the proposed method is shown in Section 4. We discuss the adversary model and usefulness, applicability, and performance of the proposed technique in Section 5. Finally, we conclude the paper in Section 5.

2. Prior Knowledge

This section describes the attack and defense techniques for the keyboard data that is the most important information in the password authentication method. The keyboard data attack techniques include the direct polling attack technique and the C/D bit vulnerability exploitation technique, while the defense technique includes a random keyboard data-generation technique.

2.1. Keyboard Data Transmission Process

A keyboard device is one of the input devices to interact with the user, supporting an interaction that instructs a command based on user input from the keyboard. The features of the keyboard are managed and processed by the operating system. This process provides features, such as key input, and shortcut keys supported by the operating system. Therefore, user data is input through the keyboard device, and delivered to the application software passed through the keyboard device stack. We depict the keyboard data transmission process in Figure 1.

The PS/2 keyboard structure consists of a key matrix and a keyboard processor inside the keyboard hardware, and a keyboard controller inside the host, a host processor, a Programmable Interrupt Controller (PIC) or Advanced Programmable Interrupt Controller (APIC) inside the host processor, and device drivers within the PS/2 keyboard device stack inside the operating system, and application programs.

When a user inputs a key using the PS/2 keyboard, the keyboard processor inside the keyboard device extracts the scancode for the key input by the user through the key matrix, and transmits it to the keyboard controller in the host. The keyboard controller receives the scancode, and sends it to PIC/APIC, to request an interrupt service. PIC/APIC is a controller for routing interrupts. Interrupt signals input from the keyboard are transferred via I/O APIC, Local PIC, etc., and cause an interrupt to the central processing unit (CPU). The CPU prepares separate tables and handlers for input and output processing, which are called the IDT and interrupt service routine (ISR), respectively.

A keyboard interrupt invokes the keyboard interrupt service routine by an entry assigned in association with the keyboard interrupt in the interrupt descriptor table described above, and then the operating system delivers the input keyboard data to the application software passed through the PS/2 keyboard device stack.

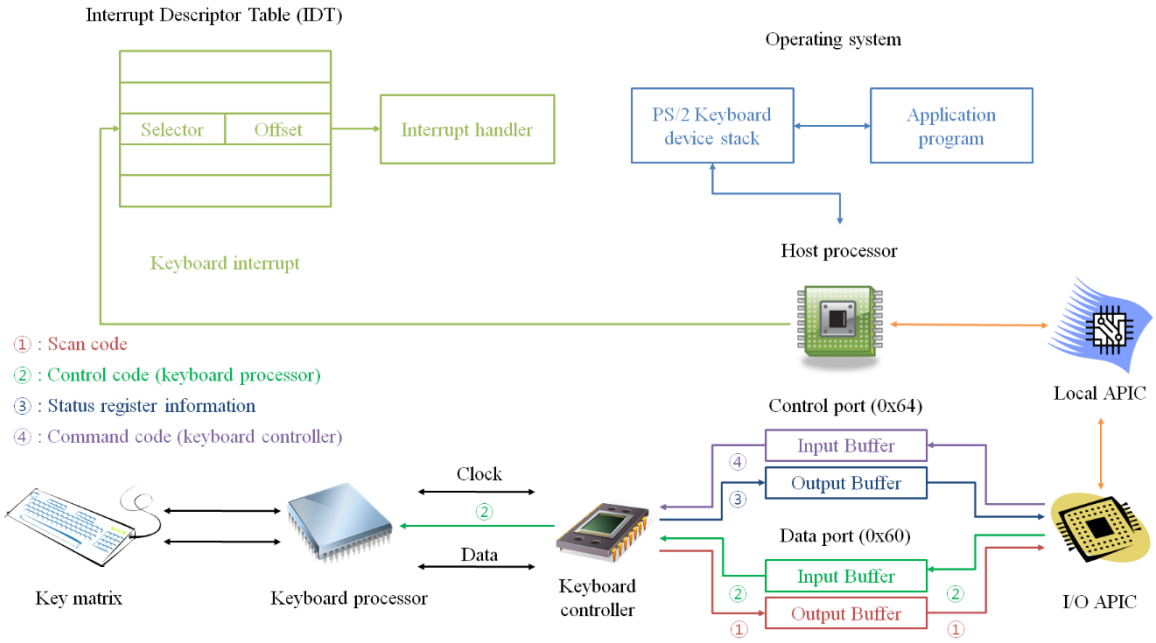

Figure 1. The keyboard data transmission process.

2.2. Keyboard Data Attack Technique Using Direct Polling

This attack technique is lower-level attack technique, which is at hardware level, than the operating system-level attacks. To input and output, Microsoft has prepared a separate atomic command (opcode, instruction) that an attacker can use to periodically check that keyboard data has been input, so that as soon as the user inputs a key, the attacker can steal keyboard data. This attack technique can have serious consequences (damage without detection), because the methods for detecting and protecting against this attack are not obvious, and the operating system also reads and processes keyboard data with the same atomic command. We experimented with the Microsoft Windows operating system. However, the proposed attack technique can attack all platforms using PS/2 keyboard and Intel processor.

For the success of this attack, an attacker must always monitor the state of the keyboard controller by reading the keyboard controller status register. Table 1 shows the information extracted from the keyboard controller status register.

Table 1. Configuration of the keyboard controller status register.

Field	Field Name
Bit 0	Output Buffer Full (OBF)
Bit 1	Input Buffer Full (IBF)
Bit 2	System flag
Bit 3	Control/Data (C/D)
Bit 4	Inhibit switch
Bit 5	Transmit time-out
Bit 6	Receive time-out
Bit 7	Parity error

For communication between the keyboard and the host, the host prepares a keyboard controller (8259A) inside the host, then sends and receives control information and data via separate ports. The control port is the 0x64 port, and the data port is the 0x60 port. Each port has its own output and input buffers for writing and reading control information. When reading the control and data ports, the status register and scancode are read, respectively. Conversely, the host sends a command to the keyboard controller or a control code to the keyboard by writing a command or control code to a control port or data port. Bits 0 and 1 of the keyboard status register represent OBF and IBF, respectively; OBF denotes that the data is filled in the output buffer, while IBF denotes that the data is filled in the input buffer. An attacker can determine if the keyboard data has been input from the keyboard by reading the control port to check whether the OBF is set. Therefore, the attacker checks if the OBF is set, and then steals the keyboard scancode by reading the data port.

2.3. Keyboard Data Defense Technique Using Random Scancode Generation

To cope with the direct polling attack, a representative defense technique is one that generates random scancodes to confuse the attacker. This technique takes advantage of the fact that the 0xD2 command in the command codes for controlling the keyboard controller provides the ability to generate a random scancode.

The defense technique generates a random scancode at any time, and then informs the keyboard controller that "I will generate a scancode", by writing a 0xD2 command to the control port. After that, if the random scancode is written to the data port, the keyboard controller raises an interrupt based on the received scancode. The interrupt passes the scancode to the security software by invoking the interrupt service routine, and the software checks whether the received scancode is the generated scancode itself. If the scancode is the random scancode generated by the security software, this means that there is no input from the keyboard. Therefore, this technique causes confusion for the attacker by repeating this random scancode generation process. If the scancode is not generated by the defender, the scancode means a scancode input from the keyboard.

Thus, this technique is a reasonably secure and effective defense technique. Even if an attacker obtains a scancode by replacing an interrupt object or using a direct polling attack technique, it is difficult for an attacker to differentiate whether it is a random scancode generated by security software or a scancode input from the keyboard, because it is impossible to classify whether the scancode is input from the keyboard or not. Figure 2 shows this defense technique using random scancode generation.

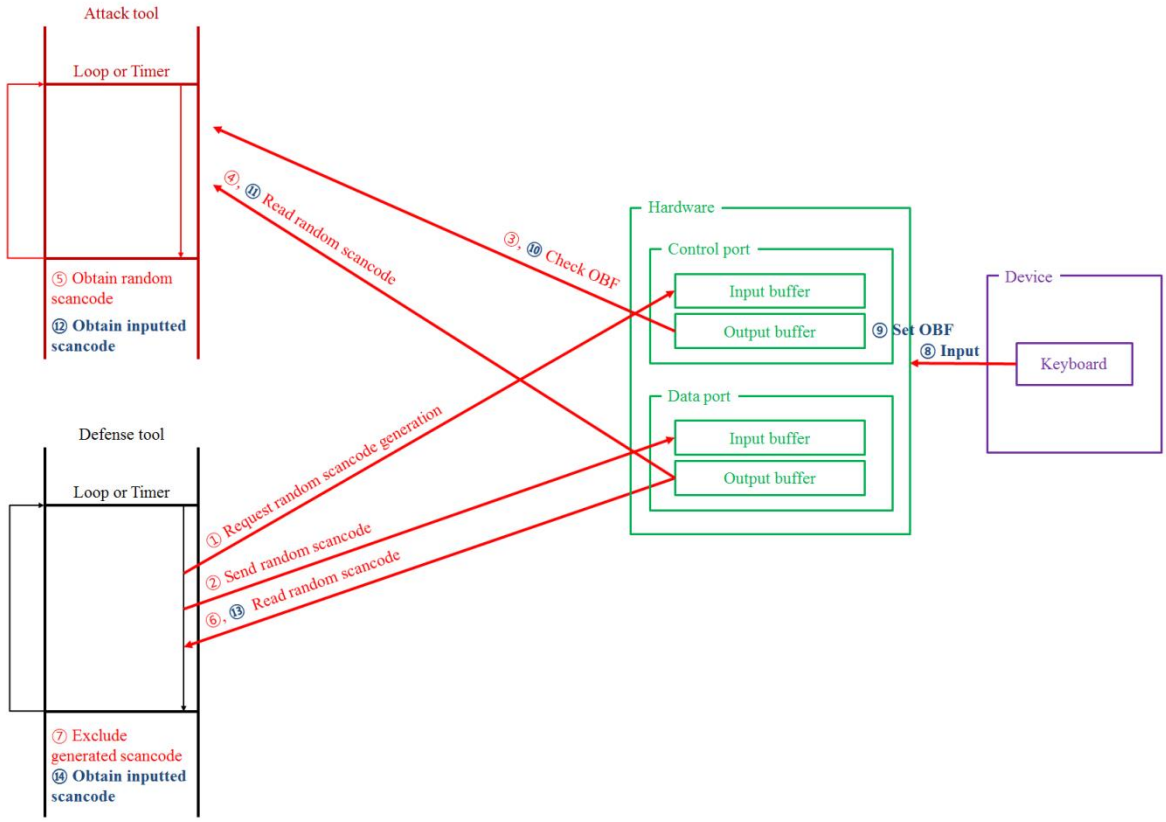

Figure 2. Operation process of the keyboard data protection technique using random scancode generation.

2.4. Keyboard Data Attack Technique Using Control/Data (C/D) Bit Vulnerability Exploitation

The defense technique described above also has vulnerability. The keyboard controller assigns a field to indicate from which port the command received from the host originates to the keyboard status register bit 3 (C/D bit) in the register. If data is received from the control port, the C/D bit is set to 1; otherwise, if data is received from the data port, the C/D bit is cleared to zero. This result is characterized by the C/D bit of the status register forming a falling edge. Therefore, an attacker determines that the scancode is a random scancode generated by the defender when OBF is set to 1, i.e., a scancode is input, after the C/D bit is set to 0 by monitoring the C/D bit. Otherwise, if the OBF is set after the C/D bit is set to 1, the scancode is determined to be the scancode input by the user, that is, the scancode is input from the keyboard, not from the defender.

By repeating this attack process, an attacker can classify all scancodes generated by the defense tool and input by the keyboard. However, to succeed in this attack, this technique requires periodic setting to check the C/D bit, and requires the condition that all input keyboard data should be preempted and collected. This causes an overload on the system and abnormal behavior and access. Therefore, there is a risk of being detected.

For this reason, in this study, we derived the security threat and the vulnerability of keyboard data by using machine learning with existing attack techniques that do not cause abnormal behavior and access, and derived the cybersecurity threat of the keyboard data. To achieve this, we proposed an attack system, and keyboard data is collected from the configured system, and then constructed datasets for using machine-learning models. In addition, we defined features to obtain only random data between collected random data and actual keyboard data, and demonstrated the practicality of classifying keyboard data based on various machine-learning models.

3. Proposed Attack System and Dataset Configuration

We describe the configuration of the proposed attack system in this section. The attack system collects all the keyboard scancodes that are input while the keyboard data defense tool is running. Moreover, we described the dataset configuration for the experiment based on the collected keyboard data using the proposed attack system.

3.1. Attack System Configuration

Figure 3 shows the keyboard data attack system proposed in this paper. The attack system collects all scancodes input from the keyboard, i.e., A1, A2 … An, while the security software generates periodically random scancodes, i.e., B1, B2 … Bn, to deceive an attacker. Consequently, the attack tool collects all scancodes, i.e., A1, B1, B2, B3, B4, B5, A2, B6, B7, A3, … , An, and Bn, input by the user and the defense tool at time of collection. Thus, the attack tool collects all the scancodes, and also collects the time when scancodes are collected. These data include the difference of the time between the current scancode and the previous scancode in nano second (ns) units. Table 2 shows examples of the collected scancodes.

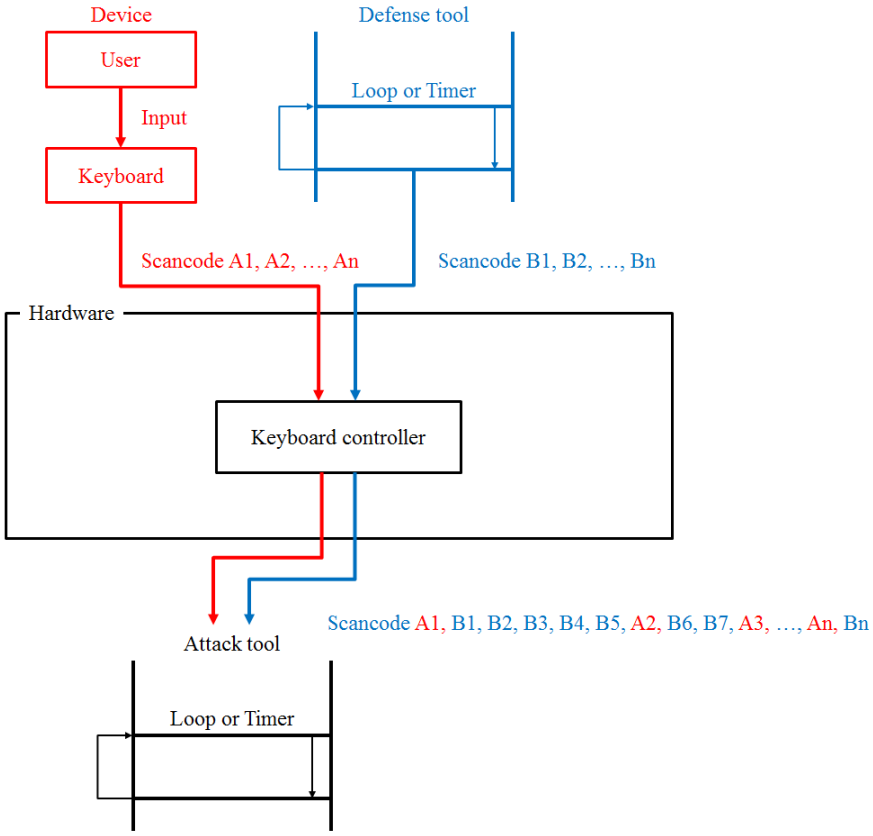

Figure 3. The proposed keyboard data attack system configuration.

Table 2. Examples of collected scancodes.

Index	Elapsed Time	Scancode	C/D Bit
65	0.062285	0x19	0
66	0.064004	0x15	0
67	0.056019	0x20	1
68	0.146008	0x23	1
69	0.186363	0x1e	0
70	0.062359	0x09	0
...	...	...	...
3503	0.062378	0x25	0
3504	0.051819	0x20	1
3505	0.154846	0x27	0
3506	0.058563	0x31	0
3507	0.062439	0x2c	0
3508	0.006104	0x1f	1
3509	0.181061	0x0b	0

The goal was to classify actual scancodes, i.e., A1, A2, ... , An, input from the keyboard from all the scancodes, i.e., A1, B1, B2, B3, B4, B5, A2, B6, B7, A3, ... , An, and Bn, collected by the attack system. To achieve this classification, we use machine learning models for using the scikit-learn library, such as k-Nearest Neighbors (KNN) [8], logistic regression [9], linear Support Vector Classifier (SVC) [10], decision tree [11], random forest, gradient boosting regression tree [12], support vector machine (SVM) [13], and multiple perceptrons (MLP) [14]. KNN labels classes with many neighbors based on the number of neighbors and then classifies data according to decision boundaries. Linear models deals with classification using linear functions. We used the logistic regression and linear support vector machine (LinearSVC) models in this study for the linear models. Decision Tree divides the data according to a yes or no question and repeats the questions until a decision is reached. Combined several machine earning models to improved performances effectively are called ensemble models, such as random forest and gradient boosting regression tree. The kernel technique includes a kernel SVM model that classifies decision boundaries according to the each data for training. This model then measures and classifies distances from the data points located at the boundaries. Finally, the neural network utilizes MLP. Moreover, three datasets were created, and the practicality of the actual keyboard data exposure was verified.

3.2. *Dataset Configuration*

To classify the real keyboard data input by the keyboard, data was collected three times to construct datasets. Table 3 shows the configured dataset.

Table 3. Whole collected data and dataset configuration.

Dataset	Total Collected Scancodes	Number of Benign Scancodes	Percentage of Benign Scancodes	Number of Malignant Scancodes	Percentage of Malignant Scancodes	Used Feature
1	3522	392	11.13%	3129	88.86%	Index, scancodes
2	10,022	1422	14.19%	8599	85.80%	Elapsed time, scancodes
3	15,046	2281	15.16%	12,764	84.83%	Elapsed time, scancodes, flag

We configured three datasets. Datasets 2 and 3 collected more than 10,000 scancodes, while Dataset 1 collected a relatively small number of 3522 scancodes. A benign scancode in the dataset refers to the actual scancode input from the keyboard, while a malignant scancode refers to the random scancode generated by the security software. The number of benign scancodes in each dataset were (392, 1422

and 2281), while the number of malignant scancodes were (3129, 8599 and 12,764), respectively. The percentages are (11.13%, 14.19% and 15.16%) for benign scancodes, respectively, and (88.86%, 85.80% and 84.83%) for malignant scancodes, respectively.

For the experiment, we configured the features in three ways. We defined the collected scancodes and indexes as the first feature, while the collected elapsed time and scancodes are defined as the second feature. Finally, the third feature is the collected scancodes, the elapsed time, and the flag (C/D). Therefore, the dataset was used for three experiments using the three datasets in three ways.

4. Experimental Results

4.1. Experiment Results Based on Feature 1 (Index and Scancode)

We classified into any number of the training set, validation set, and test set to avoid overfitting and underfitting for the experiment, and Table 4 shows experiment results of the training set, validation set, test set, and the cross-validation using Dataset 1.

Table 4. Experiment results of training set, validation set, test set, and cross-validation using Dataset 1.

Model	Parameters	Training Set Score	Validation Set Score	Test Set Score	Cross-Validation Score
KNN	n_neighbors = 4	0.90	0.88	0.88	0.894
Logistic Regression	C = 0.00001, L1 regularization	0.90	0.88	0.88	0.895
Linear SVC	-	0.90	0.88	0.21	0.773
Decision tree	max_depth = 1	0.90	0.88	0.88	0.895
Random Forest	n_estimators = 2	0.93	0.86	0.85	0.866
Gradient Boosting	learning_rate = 0.1, max_depth = 3	0.90	0.88	0.88	0.886
SVM (Support Vector Machine)	C = 1	0.92	0.88	0.88	0.894
MLP (multi-layer perceptron)	max_iter = 0.00001, alpha = 0.00001	0.90	0.88	0.88	0.746

Specifically, the training set had the best score for random forest at 0.93, while the rest of the models had a score at 0.90. The validation set had the worst score of random forest at 0.86, while the rest of the models had a score at 0.88. The test set had the worst score for linear SVC at 0.21, while the rest of the models had a score at 0.88. Cross-validation had the worst score for MLP at 0.746, while the rest of the models had a score at 0.86, except for linear SVC.

Figure 4 shows performance evaluation results of the real keyboard data input from the keyboard device according to Datasets 1 to 3. Cross-validation splits the data repeatedly, and trains multiple models. Accuracy denotes the number of correctly predicted numbers (true positive and true negative), and precision denotes the number of true positive among the number of true positive and false positive. Recall means the number of true positive among the number of true positive and false negative, and F1-score means the harmonic average of precision and recall. Area Under the Curve (AUC) is a summary of the Receiver Operating Characteristics (ROC) curve, and the AUC score falls between the worst 0 value and the best 1 value.

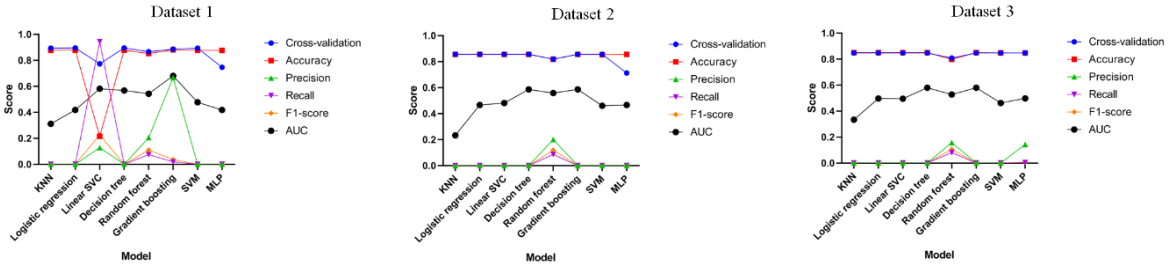

Figure 4. Performance evaluation results of cross-validation, accuracy, precision, recall, f1-score, and AUC according to Datasets 1 to 3.

To be more specific about the results for each dataset, gradient boosting had high performance in Dataset 1, while random forest had high performance in Datasets 2 and 3. Nevertheless, most of the models could not measure precision, recall, and F1-score. KNN, logistic regression, decision tree, SVM, and MLP models could not measure in Dataset 1. In Dataset 2, KNN, logistic regression, linear SVC, decision tree, gradient boosting, SVM, and MLP models could not measure, and KNN, logistic regression, linear SVC, decision tree, gradient boosting, and SVM models cannot measure in Dataset 3. In other words, a dataset consisting of index and scancode as a feature cannot classify scancode input from the keyboard and generated random scancode, which means that an attacker is very unlikely to succeed in a password stealing attack.

4.2. Experiment Results Based on Feature 2 (Elapsed Time and Scancode)

We classified into any number of the training set, validation set, and test set, Table 5 shows experiment results of the training set, validation set, test set, and the cross-validation using Dataset 1.

Table 5. Experiment results of training set, validation set, test set, and cross-validation using Dataset 1 with elapsed time.

Model	Parameters	Training Set Score	Validation Set Score	Test Set Score	Cross-Validation Score
KNN	n_neighbors = 3	0.97	0.96	0.96	0.962
Logistic Regression	C = 10000, L2 regularization	0.97	0.96	0.95	0.967
Linear SVC	-	0.90	0.89	0.90	0.895
Decision tree	max_depth = 3	0.97	0.96	0.95	0.968
Random Forest	n_estimators = 14	1.00	0.96	0.95	0.969
Gradient Boosting	learning_rate = 0.1, max_depth = 2	0.98	0.96	0.95	0.966
SVM	C = 100	0.97	0.95	0.95	0.964
MLP	max_iter = 1000, alpha = 0.00001	0.96	0.95	0.95	0.963

Experiment results of training set, validation set, test set, and cross-validation using Dataset 1.

As a result, the training set had the best score for random forest at 1.0, the rest of the models had similar score at 0.97 and above 0.90. The validation set had the worst score for linear SVC at 0.89, the rest of the models had a score at 0.96. The test set had the worst score for linear SVC at 0.90, while the rest of the models had a score at 0.95. Cross-validation had the worst score for linear SVC at 0.895, while the rest of the models had a score at 0.96.

When comparing the dataset with the index and scancode described in Section 4.1 to the dataset with the elapsed time described in this section, the random forest increased from 0.93 to 1.0, the rest of the models increased from 0.90 to 0.91 in the training set score. In the validation set score, linear SVC increased from 0.86 to 0.89, the rest of the models increased from 0.88 to 0.96. In the test set score,

linear SVC increased from 0.21 to 0.90, and the rest of the models increased from 0.88 to 0.95. In the cross-validation score, the score 0.746 at MLP increased to the score 0.895 at linear SVC, and the rest of the models increased from 0.86 to 0.96. Therefore, the all scores of performance evaluation for datasets with the elapsed time are higher than those of datasets without the elapsed time. This indicates high performance when including the elapsed time. In other words, by utilizing a dataset with the elapsed time, the real keyboard data input by the user can be classified more effectively. Figure 5 shows the detailed performance evaluation results, such as cross-validation, accuracy, precision, recall, F1-score, and AUC according to Datasets 1 to 3 with the elapsed time.

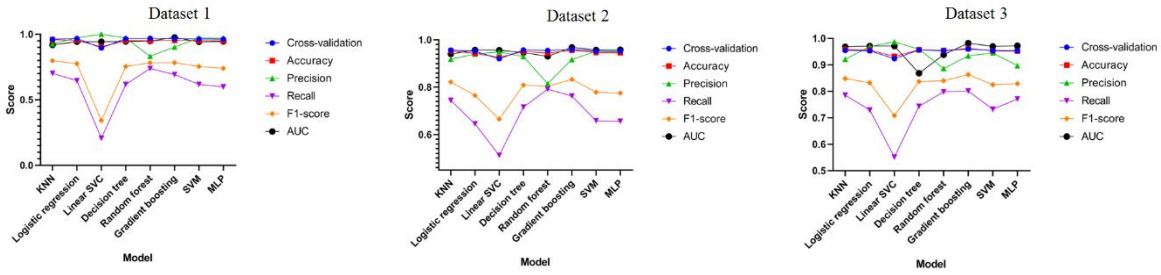

Figure 5. Performance evaluation results according to Datasets 1 to 3 with the elapsed time.

Specifically, Datasets 1 and 2 had low performance in linear SVC and random forest, while Dataset 3 had low performance in linear SVC, decision tree, and random forest. Moreover, the rest models had similar level of performance. Above all, all models evaluated the performance in all models for a dataset that includes the elapsed time when all datasets with elapsed time are compared to datasets without elapsed time. Consequently, feature 2 is appropriately selected to classify benign from malignant.

4.3. Experiment Results Based on Feature 3 (Elapsed Time, Scancode, and Flag)

Finally, we compared the proposed method with the attack technique using C/D bit that steals the keyboard data, although this attack technique causes the overload on the system, and abnormal behavior and access. We constructed a dataset containing the flag included in the attack technique using C/D bit, and analyzed the experimental results. We classified into any number of the training set, validation set, and test set, Table 6 shows experiment results of each set and the cross-validation using Dataset 1.

Table 6. Experiment results of training set, validation set, and test set scores using Dataset 1 with elapsed time and flag.

Model	Parameters	Training Set Score	Validation Set Score	Test Set Score	Cross-Validation Score
KNN	n_neighbors = 1	1.0	1.0	1.0	1.0
Logistic Regression	C = 0.01, L1 regularization	1.0	1.0	1.0	0.895
Linear SVC	-	1.0	1.0	1.0	1.0
Decision tree	max_depth = 1	1.0	1.0	1.0	1.0
Random Forest	n_estimators = 1	1.0	1.0	1.0	0.998
Gradient Boosting	learning_rate = 0.01, max_depth = 1	1.0	1.0	1.0	1.0
SVM	C = 1	1.0	1.0	1.0	1.0
MLP	max_iter = 100, alpha = 0.00001	1.0	1.0	1.0	1.0

As a result, except for special cases, all scores of all machine-learning models were close to 1.0. This means that using flag (C/D bit), it classifies the actual keyboard data effectively and completely. Figure 6

shows the detailed performance evaluation results, such as cross-validation, accuracy, precision, recall, F1-score, and AUC according to Datasets 1 to 3 with the elapsed time and flag.

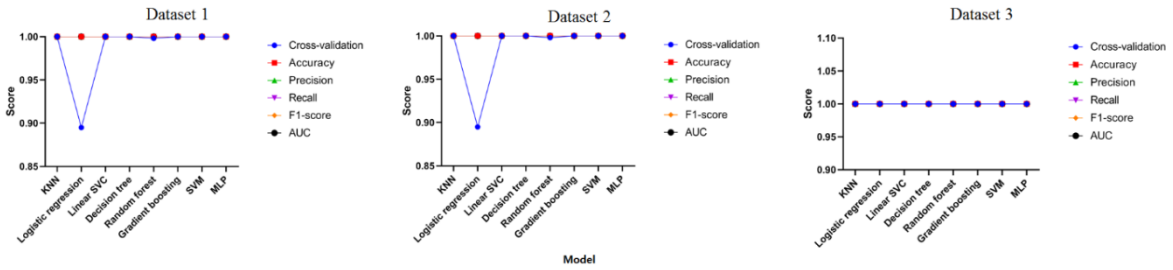

Figure 6. Performance evaluation results according to Datasets 1 to 3 with the elapsed time and flag.

Specifically, except for the logistic regression of Datasets 1 and 2 and the cross-validation score of the random forest, all models showed performance measures of accuracy, precision, recall, F1-score, and AUC has perfect score of 1.0. Specifically, if the flag is used, the data input from the actual keyboard can be classified, which means that the password of the user can be stolen easily.

4.4. Comparison of Performance Evaluation Results by Features

In this study, we evaluated the performance of datasets with index and scancode, datasets with elapsed time and scancode, and datasets with elapsed time, scancode, and flag. We demonstrated that the performances of the dataset with elapsed time and scancode and the dataset with elapsed time, scancode, and flag are higher than that of the dataset with index and scancode. Furthermore, the dataset with the flag causes system overload and abnormal behavior and access, but the proposed method effectively classifies the real keyboard data, without causing this drawback. Consequently, the results of the performance evaluation differ based on features, thereby analyzing the feature benefit by comparing the performance evaluation accordingly. Figure 7 shows the comparison of performance evaluation results of each set and the cross-validation.

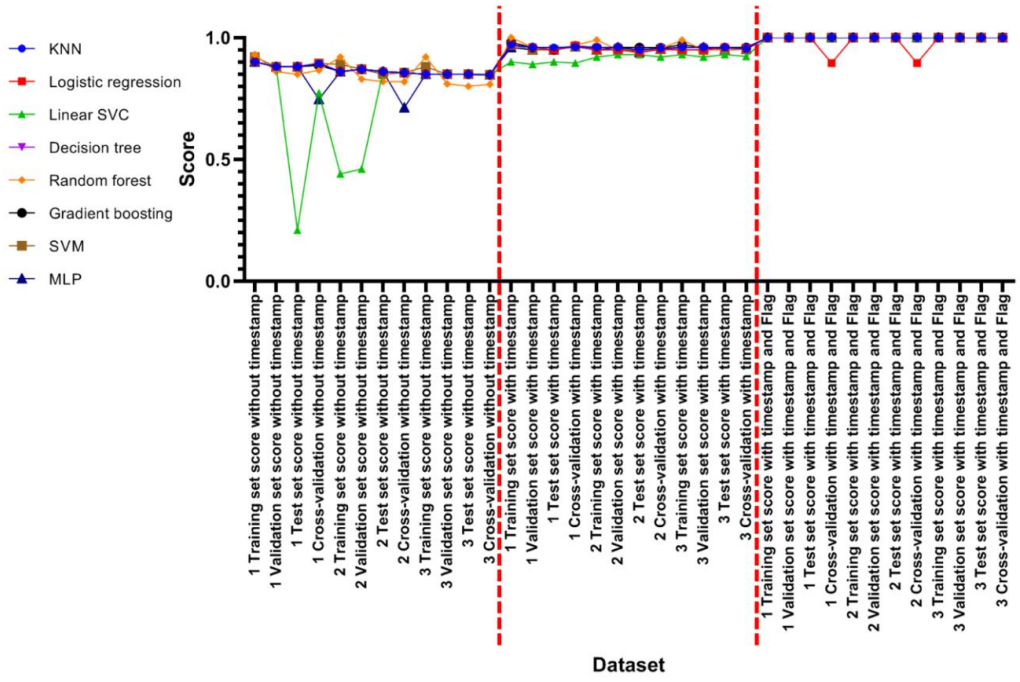

Figure 7. Comparison of performance evaluation according to each set and cross-validation by features.

In the figure, the left side is the result of the dataset with index and scancode, the middle is the result of the dataset with elapsed time and scancode, and the right side is the result of the dataset with elapsed time, scancode, and flag. The results show that the performance tends to be higher toward the right, and the dataset with elapsed time and scancode is significantly higher than the dataset with index and scancode. Moreover, performance evaluation value was closer to 1, which means that collecting data and constructing features according to the proposed method lead to performance improvement.

Analyzing the results of changes in each set and cross-validation showed significant changes. All machine-learning models using the datasets without elapsed time showed significant changes. MLP, linear SVC, and random forest showed marked differences. Conversely, all machine learning models using the datasets with elapsed time had relatively small changes, however, Linear SVC and Random Forest show significant changes.

As shown in Figure 7, performance evaluation results were significantly different depending on the features, and we verified that the dataset with the elapsed time had high performance. Figure 8 shows the comparison results for more practical performance evaluation, such as accuracy, precision, recall, F1-score, and AUC.

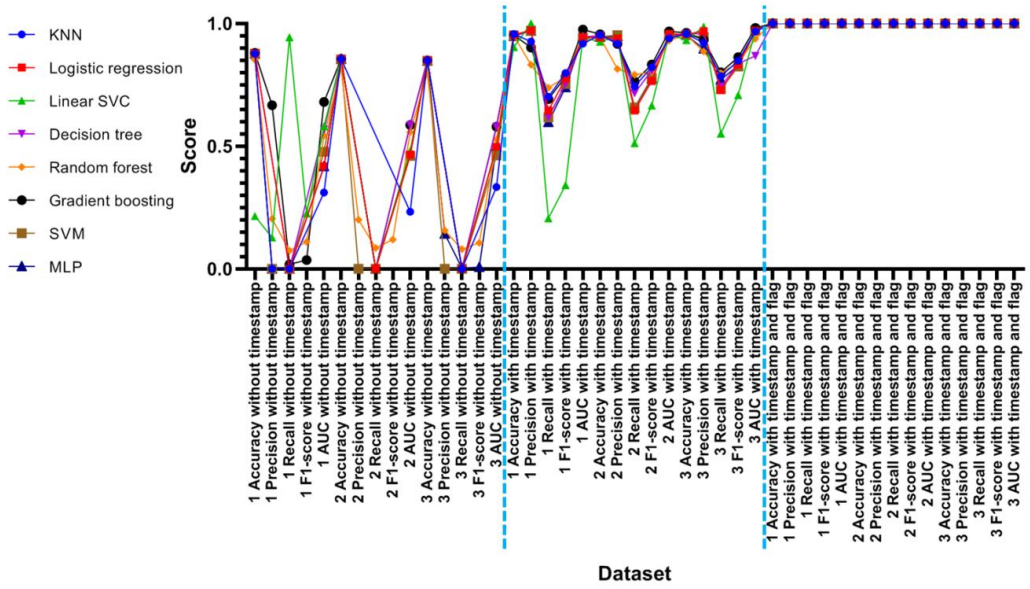

Figure 8. Comparison of performance evaluation according to practical performance evaluation by features.

In the figure, on the left is the result of the dataset with index and scancode, in the middle is the result of the dataset with elapsed time and scancode, and on the right is the result of the dataset with elapsed time, scancode, and flag. The performance tends to be higher toward the right, and the performance of the datasets with elapsed time and scancode is significantly better than the datasets with index and scancode. Moreover, the performance of the dataset is higher from 1 to 3, and the accuracy of the dataset with elapsed time and scancode is close to 1. An accuracy of close to 1 means that most passwords can be obtained by effectively differentiating between random scancodes and real scancodes.

In terms of practical performance evaluation changes, most models using the datasets without elapsed time changed considerably. Among them, gradient boosting, linear SVC, and random forest showed notable differences. Conversely, all models using the datasets with elapsed time had relatively slight changes, i.e., linear SVC, decision tree, and random forest show significant differences.

Finally, to analyze the change rate of the increase and decrease according to features, the performance differences of all datasets were analyzed shown in Table 7.

Table 7. Change rate of increase and decrease according to feature (Dataset 1).

Model	D	AC	+−	P	+−	R	+−	F	+−	AU	+−
KNN	w/o	0.877	109.1	0	-	0	-	0	-	0.311	**295.1**
	w	0.957		0.926		0.701		0.798		0.918	
Logistic Regression	w/o	0.879	108.6	0	-	0	-	0	-	0.418	225.5
	w	0.955		0.972		0.645		0.775		0.943	
Linear SVC	w/o	0.215	**420.4**	0.128	**781.2**	0.944	**21.8**	0.226	**150.8**	0.581	162.3
	w	0.904		1		0.206		0.341		0.943	
Decision Tree	w/o	0.879	**108.1**	0	-	0	-	0	-	0.568	166.3
	w	0.951		0.971		0.617		0.754		0.945	
Random Forest	w/o	0.852	111.5	0.205	405.8	0.075	984	0.11	710.9	0.543	174.7
	w	0.95		0.832		0.738		0.782		0.949	
Gradient Boosting	w/o	0.88	108.2	0.667	**135.2**	0.019	**3642.1**	0.036	**2175**	0.68	**143.3**
	w	0.953		0.902		0.692		0.783		0.975	
SVM	w/o	0.877	108.4	0	-	0	-	0	-	0.477	197.4
	w	0.951		0.971		0.617		0.754		0.942	
MLP	w/o	0.876	108.3	0	-	0	-	0	-	0.418	226.0
	w	0.949		0.97		0.598		0.74		0.945	

D: Data set. w/o: without Timestamp. w: with Timestamp. AC: accuracy. P: Precision. R: Recall. F: F1-score. AU: AUC. +−: Increase and decrease rates.

Specifically, the best model in terms of accuracy in Dataset 1 was linear SVC which increased by 420.4%, while the worst model was decision tree which increased by 108.1%. The best model in precision was linear SVC increased by 781.2%, while the worst model was gradient boosting which increased by 135.2%. The best model in recall in was gradient boosting which increased by 3642.1%. On the other hand, recall showed that linear SVC model suffered poor evaluations of Datasets with elapsed time, which decreased to 21.8%. The best model in F1-score was gradient boosting which increased by 2175%, while the worst model was Linear SVC increased by 150.8%. The best model in AUC was KNN increased by 295.1%, while the worst model was gradient boosting increased by 143.3%. Except for the linear SVC model in recall, all performance results were increased in Dataset 1, and the highest increase rate was 3642.1%.

In conclusion, using the dataset with elapsed time, we can effectively classify random keyboard data with up to 96.2% accuracy. This means that an attacker steals the real keyboard data input by the user in the real world. Consequently, the proposed attack technique discussed in this paper has derived a security threat and a new vulnerability that effectively steal user authentication information in password authentication.

5. Discussion

5.1. Adversary Model

We assumed that an attacker penetrates the victim's terminal and installs malicious programs. These are reasonable assumptions given that malicious codes continue to emerge and increase the number of zombie PCs. In this attack situation, the attacker's level is classified into two categories: professional attacker and ignorant attacker. The professional attacker exploits C/D bit vulnerability as described in Section 2.4 to steal keyboard data. A professional attacker has a high level of knowledge of the device driver. Moreover, when the attacker uses this attack technique, it obviously steals the real keyboard data. This means that an adversary neutralizes password authentication. This attack, however, induces an overload on the system with up to 99% CPU utilization, and can be detected as a malicious code by judging according to abnormal behavior and access.

An attacker using the attack technique discussed in this paper is assumed to be an ignorant attacker. An ignorant attacker is assumed to have simple programming skills at the application level or the ability to use publicly available attack tools such as keyloggers. Therefore, as described in Section 2.4, this attacker cannot use C/D bit vulnerability to steal keyboard data, hence an adversary does not neutralize password authentication. However, if this attack technique discussed in this paper is used, the ignorant attacker can steal the keyboard data. If the attacker can install keylogger tools that can be obtained easily online and obtain all keyboard data, the attacker obviously steals the user password. Here, we assume that the attacker can collect the keyboard scancode and elapsed time collected from installed keyloggers. In general, keyloggers have a low risk of being detected, because these tools do not overload to the system and do not cause abnormal behavior and access. Therefore, the ignorant attacker using the discussed attack technique will steal the real keyboard data, hence the adversary neutralizes password authentication.

5.2. Usefulness, Applicability, and Performance of the Proposed Technique

The usefulness of the proposed method is that, as described in Section 3.1, we constructed the attack system and verified the practicality of keyboard data exposure using machine-learning models. We collected 28,590 scancodes from three datasets and increased the attack success rate of keyboard attack by defining elapsed time and scancode as features. In particular, keyboard data collected from keyloggers installed by an ignorant attacker described in Section 5.1 cannot be used to steal passwords input from the user on the system that deployed the keyboard data defense technique described in Section 2.3. Nevertheless, if the same attacker uses the proposed attack technique, the attacker has a high attack success rate. Therefore, the proposed technique has usefulness.

If the attacker constructs the attack system shown in Figure 3, the attack succeeds. If the ignorant attacker, as described in Section 5.1, can install keyloggers and use machine-learning models, the proposed technique can be applicable. Therefore, the proposed technique means is applicable. Moreover, with regard to malware and vulnerability detection, the proposed attack method is applicable, because the method bypasses detection in the way of detecting malicious code or vulnerabilities presented by [15,16]. Specifically, this approach uses system-provided functions that are free from having vulnerabilities, so that an attack tool is not vulnerable or detected as malware.

Keyboard data collected from keyloggers installed by the ignorant attacker alone shows the attack success rate as low as 21.5% and up to 88.0%, even with machine-learning models used. On the contrary, defining elapsed time as a feature proposed in this paper, the attack success rate is as low as 90.4% and up to 96.2%. Moreover, the performances proposed in this paper by the performance evaluation measures were significantly better. Therefore, we verified that the proposed technique has high performance.

6. Conclusions

This study demonstrated the security of the keyboard data using machine learning in password authentication while the keyboard data defense tool is running. Conventional attack techniques have limitations in distinguishing between random keyboard data and real keyboard data. To overcome this problem, we collected the keyboard data to construct datasets with features and classified real keyboard data effectively. In experimental results using constructed datasets, the datasets collected by the proposed method classified the actual keyboard data significantly better than conventional attack techniques. Namely, the proposed attack technique steals keyboard data with high accuracy. Specifically, performance measures in terms of practical evaluation, such as accuracy, precision, recall, F1-score, and AUC were better than in the conventional attack techniques, and have exceedingly low false positive and false negative rates. In addition, the best accuracy was 96.2%, which means that the user input data is obviously classified. The proposed method in this paper derived a new security threat and a new vulnerability of the password authentication method. Conclusively, this study will have advantages in the security assessment of password authentication and all types of authentication technology and application services input from the keyboard.

Author Contributions: Conceptualization, K.L. and K.Y.; methodology, K.L.; software, K.L.; validation, K.L.; data curation, K.L.; writing—original draft preparation, K.L.; writing—review and editing, K.Y.; supervision, K.Y.; project administration, K.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. 2018R1A4A1025632) and the Soonchunhyang University Research Fund.

Acknowledgments: This work was partially supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2018R1A4A1025632) and the Soonchunhyang University Research Fund.

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

References

1. Conklin, A.; Dietrich, G.; Walz, D. Password-based authentication: A system perspective. In Proceedings of the 37th Annual Hawaii International Conference on System Sciences, Big Island, HI, USA, 5–8 January 2004.
2. Lee, H.; Lee, Y.; Lee, K.; Yim, K. Security Assessment on the Mouse Data using Mouse Loggers. In Proceedings of the International Conference on Broadband and Wireless Computing, Communication and Applications, Asan, Korea, 5–7 November 2016.
3. Lee, K.; Yim, K. Keyboard Security: A Technological Review. In Proceedings of the 2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Seoul, Korea, 30 June–2 July 2011.
4. Lee, S.; Lee, K.; Yim, K. Security Assessment of Keyboard Data: Based on Kaspersky Product. In Proceedings of the International Conference on Broadband and Wireless Computing, Communication and Applications, Asan, Korea, 5–7 November 2016.
5. Lee, K.; Bae, K.; Yim, K. Hardware Approach to Solving Password Exposure Problem through Keyboard Sniff. *Int. J. Electr. Comput. Eng.* **2009**, *3*, 1501–1503.
6. Yim, K. A new noise mingling approach to protect the authentication password. In Proceedings of the 2010 International Conference on Complex, Intelligent and Software Intensive Systems, Seoul, Korea, 30 June–2 July 2012.
7. Lee, K.; Yim, K. Password Sniff by Forcing the Keyboard to Replay Scan Codes. In Proceedings of the 6th Joint Workshop on Information Security, Kaohsiung, Taiwan, 5–6 October 2011.
8. Zhang, M.L.; Zhou, Z.H. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognit.* **2007**, *40*, 2038–2048. [CrossRef]
9. Cheng, W.; Hüllermeier, E. Combining instance-based learning and logistic regression for multilabel classification. *Mach. Learn.* **2009**, *76*, 211–225. [CrossRef]
10. Schlkopf, B.; Smola, A. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*; MIT Press: Cambridge, MA, USA, 2001.
11. Sinclair, C.; Pierce, L.; Matzner, S. An application of machine learning to network intrusion detection. In Proceedings of the 15th Annual Computer Security Applications Conference, Phoenix, AZ, USA, 6–10 December 1999.
12. Banfield, R.E.; Hall, L.O.; Bowyer, K.W.; Kegelmeyer, W.P. A Comparison of Decision Tree Ensemble Creation Technique. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *29*, 173–180. [CrossRef] [PubMed]
13. Jan, S.U.; Lee, Y.D.; Shin, J.; Koo, I. Sensor Fault Classification Based on Support Vector Machine and Statistical Time-Domain Features. *IEEE Access* **2017**, *5*, 8682–8690. [CrossRef]
14. Yin, C.; Zhu, Y.; Fei, J.; He, X. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access* **2017**, *5*, 21954–21961. [CrossRef]
15. Liu, S.; Lin, G.; Han, Q.L.; Wen, S.; Zhang, J.; Xiang, Y. DeepBalance: Deep-Learning and Fuzzy Oversampling for Vulnerability Detection. *IEEE Trans. Fuzzy Syst.* **2019**. [CrossRef]
16. Liu, S.; Dibaei, M.; Tai, Y.; Chen, C.; Zhang, J.; Xiang, Y. Cyber Vulnerability Intelligence for IoT Binary. *IEEE Trans. Ind. Inform.* **2020**, *16*, 2154–2163. [CrossRef]