Received November 1, 2019, accepted November 24, 2019, date of publication November 29, 2019, date of current version December 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2956819

Vulnerability Analysis Challenges of the Mouse Data Based on Machine Learning for Image-Based User Authentication

KYUNGROUL LEE[1], **CHRISTIAN ESPOSITO**[2], **AND SUN-YOUNG LEE**[3]

[1]R&BD Center for Security and Safety Industries, Soonchunhyang University, Asan 31538, South Korea
[2]Department of Electrical Engineering and Information Technology (DIETI), University of Napoli Federico II, 80125 Napoli, Italy
[3]Department of Information Security Engineering, Soonchunhyang University, Asan 31538, South Korea

Corresponding author: Sun-Young Lee (sunlee@sch.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) Grant Funded by the Korean Government (MSIT) under Grant 2018R1A4A1025632, the Basic Science Research Program through the National Research Foundation of Korea (NRF) Funded by the Ministry of Education under Grant NRF-2018R1D1A1B07047656, and the Soonchunhyang University Research Fund.

ABSTRACT With the change from the pre-internet era to online society, user authentication technology is required, and for that, password-based authentication technology is generally used. However, the technology has vulnerabilities and security threats that cannot ensure security and reliability, due to the exposure of the keyboard data that comprises a password input from the keyboard. In order to settle this problem, image-based authentication technology has emerged; but the password input from the mouse is not secure, due to the exposure of the mouse data. This problem has led to the emergence of mouse data protection technology. This technology protects mouse data by generating a large number of random mouse positions at any time, thereby inducing an attacker to track any mouse position generated by the defender, even if the attacker takes over the mouse data. Therefore, this mouse protection technology almost completely defends against existing mouse data attack techniques. With mouse data protection technology applied, the challenge of this paper is to verity the feasibility of mouse data attack. For the experiment, we collected both random mouse data generated by the defender and real mouse data input from the user, and verified the security of mouse data using mouse data classification based on machine learning. As a result of the experiment, we have verified the stealing of mouse data by using the proposed method with high quality, even if existing techniques of mouse data attack do not steal real mouse data. The best accuracy is 98%. In other words, the proposed method almost completely classifies the mouse data input from the user. Consequently, this paper derives and verifies the vulnerability and security threat of image-based authentication technology. Moreover, the vulnerability and security threat found in this paper not only constitute a new vulnerability and security threat, but can also be used as a criterion in security analysis and evaluation for image-based authentication technology.

INDEX TERMS Image-based authentication, user authentication, machine learning, vulnerability analysis.

I. INTRODUCTION

With the change from the pre-internet era to online society, online user authentication technology has been required. In the pre-internet era, user identity is verified face-to-face based only on documents owned by the user, such as their Social Security Number (SSN), driver's license, or passport.

The associate editor coordinating the review of this manuscript and approving it for publication was Zhen Qin.

However, in online society, there is a need for a technique that replaces user authentication, because in online society, the user does not need face-to-face identification. According to these demands, user authentication techniques for online use have emerged from the past, and a representative user authentication technique is password-based authentication [1]. Due to its ease of deployment, this technique has been introduced in a variety of applications, such as operating systems, web services, and even physical systems. More specifically, during the registration process, the user transfers a password in the form of a character string desired by the user to the service provider, and the service provider then stores the password received from the user. In the next step, the authentication process, the user requests the service provider to provide his or her desired service, and the service provider then requests the registered password from the user. The user transfers the password registered in the registration process to the service provider, and the service provider authenticates the user by comparing the password stored in the registration process with the password received in the authentication process. In this authentication technique, the important information related to the authentication is the password, which is usually passed from the keyboard connected to the user's system. Namely, the information that must be protected in password authentication is keyboard data input from the keyboard.

Nevertheless, the problem of password authentication has been revealed by neutralizing keyboard data protection [2]–[4]. The keyboard is connected via a PS/2 interface and a USB interface. However, when designed, these interfaces did not take security into account. As a result, it was impossible to protect keyboard data; and eventually, a vulnerability was found, in which the data input from the keyboard was exposed. This means that the user's password is exposed, which neutralizes password authentication. To solve the problem of keyboard data exposure, keyboard security technology has emerged. The process of transmitting keyboard data is as follows. When the user input a key, the keyboard collects the user-input data, which is then passed via the PS/2 or USB interface to the host. The host is preparing for an interrupt to notify the transfer of data from connected input and output devices such as the keyboard. Hence, the host provides a separate module for handling interrupts, called Programmable Interrupt Controller (PIC) and Advanced Programmable Interrupt Controller (APIC). That is, when data is transmitted from an input and output device, the PIC and APIC checks which device the request is from, and generates a corresponding interrupt to the CPU. For example, when data is passed from the keyboard, the PIC and APIC checks interrupts from the keyboard device, and generates a keyboard interrupt to the CPU. When the keyboard interrupt occurs, the operating system calls the keyboard interrupt service routine, which obtains and processes the keyboard data. When the interrupt service routine obtains the keyboard data, the data is passed to the application program via the device drive (chain), and then the application program finally receives the keyboard data input from the user. Therefore, the layers in the process of transferring the keyboard data from the keyboard to the application program are classified into a hardware layer, an operating system (kernel) layer, and an application program layer.

Most early keyboard security techniques protected keyboard data in the application program layer. However, when the attacker and a defender compete in the same layer, an attacker is advantaged. This is because the defender must protect the keyboard data while retaining the stability of system. Moreover, in order to overcome the failure of the attack caused by competing with the defender in the application program layer of the same user level, the attacker attempts to attack at the kernel level, which is lower than the user level, to steal keyboard data. To cope with such kernel level attack techniques, the defender has introduced defense techniques to protect keyboard data at the kernel level by applying techniques such as interrupt service routine replacement, and filter drivers. Nevertheless, because the attacker and the defender are competing at the same level, the attacker is also advantaged over the defender. In addition, in order to overcome the failure of attack caused by competing with the defender at the same kernel level, the attacker attempts to attack at the hardware level, which is lower than the kernel level, to steal keyboard data. In order to prevent such attacks at the hardware level, the defender has introduced defense techniques to monitor the access of hardware. This is because it is difficult for the defender to protect keyboard data from attacks at the hardware level. This technology prevents accessing 0×60 and 0×64 keyboard ports from other processes than the defense process by monitoring keyboard ports, and this prevents the exposure of keyboard data even if the attacker attempts a hardware level attack. Despite these defenders' efforts, this defense technique has the same powers as attackers and defenders. For this reason, the attacker can steal keyboard data by neutralizing keyboard surveillance technology. In other words, current technologies for protecting keyboard data have limitations, which mean that password authentication is neutralized. Consequently, a user authentication technology that is more secure than the password authentication is required; and in response, image-based user authentication technology has emerged [5].

Image-based authentication technology is an authentication technology that displays a specific image, such as a keypad, on the screen, and uses the click information on the image as a password. The information that must be protected in this technology is the image information displayed to the output device, and the mouse data information input from a mouse [6]. Therefore, unlike the password authentication technology, this technology does not receive authentication information from the keyboard, but from the mouse. This means that due to the keyboard data exposure, user authentication is prevented from being neutralized. Nevertheless, the problem of image-based authentication has been found by failing to protect the image information displayed to the output device and the mouse data input from the mouse [7], [8]. In general, the mouse uses a PS/2 interface, a USB interface, a touchpad, and a touch screen. As with the keyboard, security is not taken into account when these mouse interfaces was designed. This makes it impossible to protect the input mouse data. Moreover, vulnerabilities have been revealed due to mouse location information managed by the operating system. The operating system, especially the graphical operating system, manages the position of the mouse cursor for interaction with the user, and this handles commands from the user, for example, executing by double-clicking, moving, and selecting by clicking. In order to provide such features, the operating system must store and manage the position of the mouse cursor by itself, and transmit location information when the application program requests the position. In other words, the operating system provides APIs that allow the application to obtain the mouse position. Hence, an attacker can trace mouse movement input from the user by abusing APIs associated with the mouse cursor location provided by the operating system [9]. For example, the attacker, such as real-time active adversary [10], continuously collects mouse positions the user is moving to by periodically calling the GetCursorPos() function. This means tracking the movement of the cursor that the user commands.

In order to cope with the mouse attack technique, a mouse data protection technique that uses the SetCursorPos() function that prevents an attacker from tracking the mouse movement has emerged [11]. The key idea of this defense technique is not to expose the actual location input from the user, but to expose the mouse location information. The reason is that, as demonstrated by keyboard data attack and defense techniques, it is difficult to prevent access to mouse data. In other words, the defender generates a random mouse position to forcibly move the mouse cursor, and by filtering the generated mouse positions, protects the mouse position input from the actual user. The reason is that because the defender knows the random mouse position, the defender classifies the real data and the dummy data by recognizing dummy data, not the mouse data input from the user. On the other hand, by calling the GetCursorPos() function, the attacker collects both random and actual mouse positions generated by the defender at the same. However, the attacker hardly recognizes whether the collected mouse position is from the user, or from the defender. This means that the attacker does not succeed in the attack.

By using the mouse data attack and defense techniques as described above, mouse data protection techniques, such as keyboard data attack and defense techniques, have the result of protecting or exposing mouse data in various ways at various hierarchical layers. Here, the security of the mouse data is closely related to the security of the image-based authentication technology. Therefore, in this paper, we have challenges to derive the vulnerability of mouse data by analyzing the feasibility of the attacker's exploitation of mouse data practically in the situation where the mouse data protection technique is applied. By doing so, the attacker must classify any mouse position data generated by the security software and mouse data input from the user. Once the two types of data are classified, the attacker can trace mouse movements. Nevertheless, as mentioned above, calling APIs periodically, such as the GetCursorPos() function, is not enough to steal the actual mouse data input from the user.

In this paper, we propose a method for classifying random mouse cursor locations and mouse data input from the user for the security analysis of mouse data. To do this, we analyze the feasibility of the classification of mouse data by using machine learning based on the attack technique using the GetCursorPos() function that uses absolute coordinates. This is because an attack technique using the WM_INPUT message has the problem that error occurs between absolute coordinates and relative coordinates. In order to solve this problem, the attacker uses the attack technique that uses the GetCursorPos() function, which uses the same mouse data as a reference with the defense technique, in order to reduce the error of the mouse movement. As described above, the conventional mouse attack technique makes it difficult for the attacker to steal mouse positions input by the actual user using the GetCursorPos() function, when the defender generates random mouse positions using the SetCursorPos() function. In other words, the attack using the GetCursorPos() function does not steal mouse data. Nevertheless, we verify the feasibility of classifying the user's mouse coordinates by using machine learning, based on all collected coordinates.

The contribution of this paper is as follows:

- Existing mouse data attack techniques that utilize the GetCursorPos() function do not neutralize defense techniques that utilize the SetCursorPos() function. For this reason, in this paper, we propose a method to classify random mouse data based on machine learning for attacks using the GetCursorPos() function. We have verified that the attack using the GetCursorPos() function, which is not possible to steal the mouse data, classifies mouse data with high quality based on machine learning.
- The information that can be collected in an attack technique using the GetCursorPos() function is the X and Y coordinates. In this paper, we analyze the method by which the defender generates random mouse data as the key point, and determine that it is a form of calling random mouse data periodically, such as a timer. Therefore, datasets are constructed by collecting the X and Y coordinates and the time difference between the coordinates. As a result, we technically verify that datasets constructed by the data collection method proposed in this paper effectively classify random mouse data generated by the defender.
- Mouse data classification of existing attack techniques has low accuracy, high false positive rate, and high false negative rate. However, by constructing a feature of the time difference from which coordinates are collected to datasets, the proposed method demonstrates high accuracy, low false positive rate, and low false negative rate.
- The results of this paper derive and verify a vulnerability and the security threat of image-based authentication technology. The proposed attack technique is extremely effective and accurate. The best accuracy is 98 %, which means that the random mouse data input from the user is almost completely classified. Thus, if an attacker captures the victim's screen, and then collects and classified the mouse data, the attacker can trace the mouse movements of the user. In other words, it is possible to steal the password input from the user. This means a new vulnerability and security threat in

image-based authentication technology. Moreover, this threat is not only a new threat, but can also be used as a criterion for security analysis and evaluation for image-based authentication.

This paper is organized as follows. Section 2 discusses the mouse data transfer process and the existing mouse data attack and defense techniques, which are necessary for understanding the proposed mouse data attack technique. Section 3 describes the attack system configuration and mouse dataset and configuration collected from the proposed attack system. Section 4 presents the experiment results of mouse data attack based on the proposed technique; and finally, Section 5 concludes the paper.

II. PRIOR KNOWLEDGE

This section describes the mouse data attack and defense techniques in image-based authentication techniques. We describe representative techniques that are relatively easy for attackers to access and collect mouse data. Namely, the attacker attempting an attack proposed in this paper assumes that the attacker does not have high-level knowledge for the attack, and we verify a security threat of the exposure of the password in image-based authentication by stealing mouse data input from the user from the aspect of such an attacker. The mouse data attack technique includes an attack that uses the GetCursorPos() function, while the defense technique includes a technique that uses the SetCursorPos() function.

A. MOUSE DATA TRANSFER PROCESS

The mouse device is an input device for interacting with the user, and supports the selection of items by moving and clicking the mouse cursor. The mouse cursor is managed by the operating system, and the operating system receives and processes mouse data input from the mouse device, which is hardware. This process determines the mouse position on the screen managed by the operating system, and moves the mouse cursor. Therefore, the mouse data is input from the mouse device, which is hardware, and transferred to the application program via the device driver of the operating system. Fig. 1 shows the mouse data transfer process.

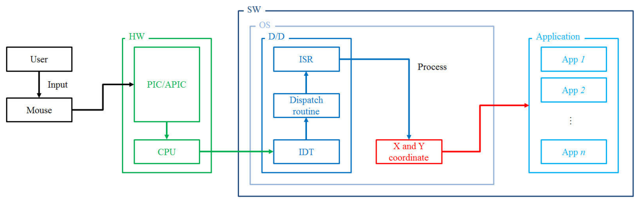


FIGURE 1. **The mouse data transfer process.**

The detailed mouse data transfer process is as follows. When the user moves the mouse device, the device generates data corresponding to the moved position, and transfers it to the host. The host prepares PIC/APIC, which is the controller for handling interrupt from devices. APIC distinguishes between interrupts from various devices connected to the host and delivers interrupt messages to the CPU to handle them correctly in the operating system. In the past, the CPU handled all input and output by itself. However, in order to improve performance and efficiency, the CPU has evolved to call and handle routines by preparing separate tables and handlers for the necessary input and output processing in advance when interrupt occurs. These tables and handlers, called interrupt descriptor tables (IDTs) and interrupt service routines (ISRs), determine the final mouse position by processing the data input from these handlers. The mouse supports a variety of modes and generally carries the relative coordinates of the distance moved. The operating system manages the mouse cursor position for interaction with the user, and moves the coordinate by the input position from the current position when relative coordinates are input from the mouse through the above process. Left and right movements are X coordinates, while up and down movements are Y coordinates. More specifically, moving to the left applies X coordinates as minus, and moving to the right applies X coordinates as plus. Moreover, moving upwards applies Y coordinates as minus, while moving downwards applies Y coordinates as plus. Consequently, the numerical value is calculated according to the distance moved, and the direction and the numerical value are added to the operating system. Finally, the mouse coordinates are passed to the requesting application program.

B. MOUSE DATA ATTACK TECHNIQUE

This section describes an attack technique using the GetCursorPos() function. This technique is a representative mouse attack technique by which an attacker can easily access and collect mouse data. Since application programs have their own UIs, they need to provide commands or features at specific locations. The operating system provides an API for passing the mouse position on the current screen to the application program, such as the GetCursorPos() function. When the application program obtains the current mouse position, the program calls this function, which as shown in Fig. 1, passes the mouse cursor position managed by the operating system. After that, the application program converts the received coordinates into coordinates inside the application program, as needed. However, by exploiting this function, the attacker can trace the mouse movements by the user, as shown in Fig. 2.

The detailed mouse attack technique is as follows. The operating system stores and manages the current mouse cursor position. An attack tool calls the GetCursorPos() function to obtain the current mouse position on the screen, and then the operating system passes the mouse position requested from the attack tool. The attack tool obtains the current mouse cursor position by calling the function once. Hence, the current mouse position is periodically collected by using a timer or loop to trace the mouse movement input from the user. By setting up this process in a very short period of time, the attacker almost completely traces the mouse movement.

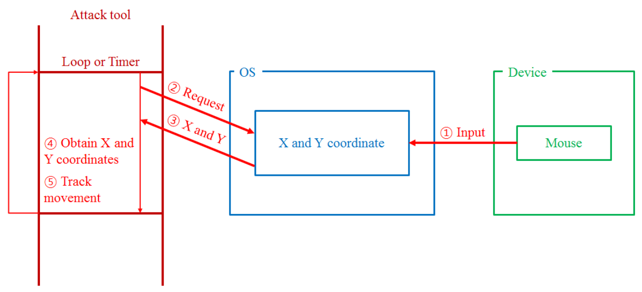
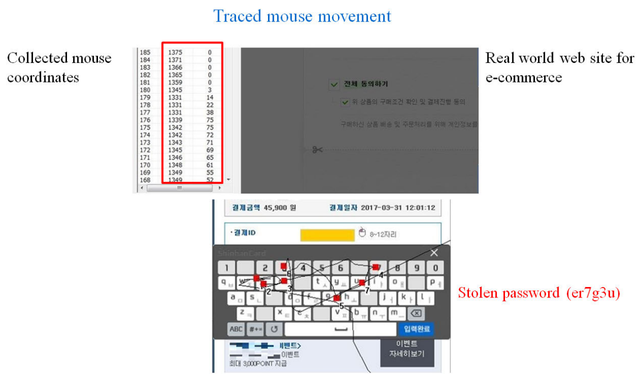


FIGURE 2. Mouse data attack technique using the GetCursorPos() function.


Traced mouse movement
Collected mouse coordinates
Real world web site for e-commerce
Stolen password (er7g3u)

185	1375	0
184	1371	0
183	1366	0
182	1365	0
181	1359	0
180	1345	3
179	1331	14
178	1331	22
177	1331	38
176	1339	75
175	1342	75
174	1342	72
173	1343	71
172	1345	69
171	1346	65
170	1348	61
169	1349	55
168	1349	52

FIGURE 3. Example of mouse movement tracing using the GetCursorPos() function.

Fig. 3 shows the traced result of mouse movement using the GetCursorPos() function.

Specifically, the top-left shows the result of collected mouse coordinates, including X and Y coordinates, and the number of coordinates collected. The bottom shows the result of stealing the password input from the user in the real-world website with image-based authentication technology based on the collected mouse data. The top-left figure consists of three columns, each representing the collected mouse data sequence number, X coordinate, and Y coordinate. When an attacker starts this attack, the attacker captures the image on the screen. After that, the mouse coordinates are collected in the process as shown in Fig. 2, and the collected coordinates are displayed to the captured image in real time. As a result, as shown in the figure below, the user's mouse movement can be tracked, and the input password can be stolen. We experimented by applying the image-based authentication technique to e-commerce web sites in South Korea, and this attack technique verified that the password was exposed, even for international web sites.

C. MOUSE DATA DEFENSE TECHNIQUE

This section describes a defense technique using SetCursorPos() function, a representative mouse defense technique corresponding to the mouse data attack technique described in Section 2.2. Due to the attack using the GetCursorPos() function, it does not prevent the tracking of the user's mouse movement. Moreover, the GetCursorPos() function is a legal

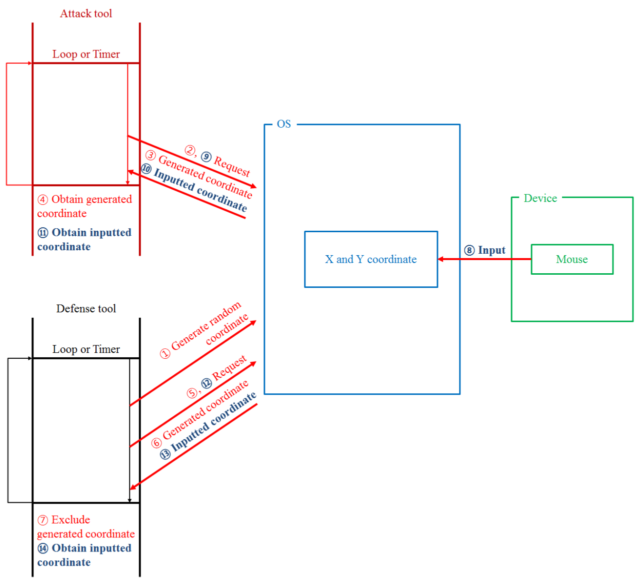


FIGURE 4. Mouse data defense technique using the SetCursorPos() function.

API provided by the operating system, so it is difficult to determine that a program is a malicious code or attack only by detecting the calling of this function. In terms of attack detection, defending a user level attack at the user level is difficult to prevent effectively, because an attacker and a defender are in competition with each other. In addition, the defender has a limitation to applying relatively strong or illegal techniques to the attacker, because the defender must defend by supporting successful operations, to avoid errors and bugs. In order to overcome this limitation, a defense technique using SetCursorPos() function, which is a technique to prevent tracking the mouse position input from the user even when the mouse cursor position is stolen, has appeared, rather than detecting or preventing calling the GetCursorPos() function.

This defense technique is a trick technique that induces an attacker to obtain the incorrect mouse positions generated by a defender who generates random cursor positions at any time, even if the attacker steals the over-all mouse positions. Since the attacker does not distinguish random coordinates, it is almost impossible to track the mouse position input from the user. On the other hand, if the defender filters the generated coordinates from the collected mouse coordinates, the defender can correctly obtain the mouse coordinates input from the user, because the defender knows the random coordinates. That is, in image-based authentication, it is possible to authenticate by clicking a location corresponding to a password by the user moving the mouse successfully. Fig. 4 shows the defense technique process using the SetCursorPos() function.

The defense tool generates random coordinates to induce the attacker, and then manipulates the mouse cursor position with the random coordinates by calling the SetCursorPos() function. The attack tool calls the GetCursorPos() function to obtain the current mouse position, and the operating sys-

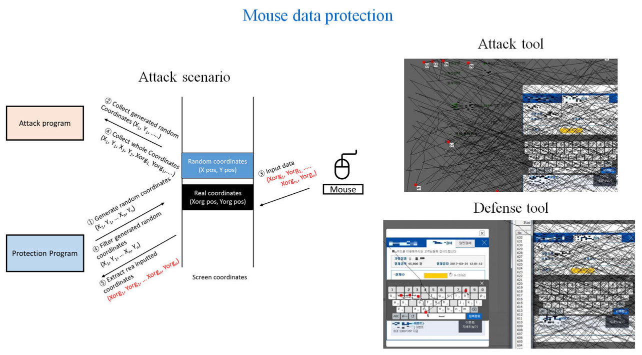

Mouse data protection
Attack scenario
Attack program
Protection Program
② Collect generated random Coordinates (X_1, Y_1, ...)
④ Collect whole Coordinates (X_1, Y_1, X_2, Y_2, $Xorg_1$, $Yorg_1$,....)
① Generate random coordinates (X_1, Y_1, ... X_n, Y_n)
④ Filter generated random coordinates (X_1, Y_1, ... X_n, Y_n)
⑤ Extract rea inputted coordinates ($Xorg_1$, $Yorg_1$, ... $Xorg_n$, $Yorg_n$)
Random coordinates (X pos, Y pos)
Real coordinates (Xorg pos, Yorg pos)
Screen coordinates
③ Input data ($Xorg_1$, $Yorg_1$, ... $Xorg_n$, $Yorg_n$)
Mouse
Attack tool
Defense tool

FIGURE 5. Example of mouse data protection result using the SetCursorPos() function.

tem passes the mouse position requested by the attacker. At this time, the attack tool can successfully obtain the mouse position, but the position is the random coordinate generated by the defense tool, in order to trick the attacker. After that, when the user moves the mouse, the mouse transmits the coordinates corresponding to the movement to the operating system, and the operating system updates the current mouse position by applying the coordinates transmitted from the mouse.

After the mouse input is completely by the user, the attacker also calls the GetCursorPos() function to steal the current mouse position. At this point in this process, the attacker cannot distinguish between the coordinates generated by the defense tool, and the coordinates input from the mouse. On the other hand, if the collected coordinate is the same as the generated coordinate, the defense tool filters the coordinate, because the tool knows the coordinate it generated. The other coordinates are displayed as mouse movements to the screen to interact with the user to show inputting the password. Fig. 5 shows the result of mouse data protection using the SetCursorPos() function.

In detail, the left side of the above figure shows an attack scenario in which an attacker attempts to steal mouse data using the GetCursorPos() function, while protecting the mouse data using the SetCursorPos() function. The top-right shows the result of the stolen mouse data by the attacker based on this attack scenario. The results collect not only mouse data input from the user, but also random mouse data generated by the defender. Therefore, the attacker does not filter the mouse data input from the user, which means that the attacker does not steal the password of the user in the image-based authentication technique.

On the other hand, the defender correctly filters and classifies the mouse data input from the user; this allows the user to input a password in image-based authentication, as shown at bottom-right. When the user moves the mouse position over the keypad displayed on the screen, the defense tool filters out any mouse data generated by the tool, and displays only the mouse data input from the user. Accordingly, the user inputs the password by moving to the character corresponding to the password registered by the user, and by sequentially clicking the password. In the figure, the red square is the position where the user clicks. Consequently, the mouse data protection technique using the SetCursorPos() function effectively prevents the mouse data attack technique using the GetCursorPos() function.

As described above, a defender protects the mouse data by applying the mouse data defense technique. In this paper, we attempt to verify the feasibility of mouse data stealing by an attacker under this defense. If the attacker attempts a high level of attack techniques, such as a hardware access-based attack and a kernel-based attack, mouse data can be stolen. However, it takes a lot of effort and time to attempt these attack techniques, and these attacks have the risk of being detected due to abnormal approaches. Therefore, in this paper, we derive a vulnerability and a security threat of the mouse data, and verity the security by applying machine learning to the attack technique using the existing GetCursorPos() function, not the attack requiring a high level of attack technique. By doing so, this paper constructs an attack system of mouse data, and collects mouse data from the system to construct datasets. Moreover, we configure features to classify only dummy data between collected dummy data and actual mouse data, and verify the feasibility of classifying mouse data by applying various machine learning models.

III. ATTACK SYSTEM AND DATASET CONFIGURATION

This section describes the composition of the proposed attack system. The configured system applies a method for collecting all the mouse movements that are input in the situation where the mouse data defense technique is executed. In addition, we describe the dataset configuration required for the experiment based on the mouse data collected from the configured system.

A. ATTACK SYSTEM CONFIGURATION

Fig. 6 shows the proposed mouse data attack system. The attack system must collect mouse data input from the mouse device, that is, A1, A2, ..., An, by calling the GetCursorPos() function. In this process, the defense tool generates random coordinates, namely, B1, B2, ..., Bn, periodically, to trick the attacker. As a result, the attack tool obtains coordinates input from the user and coordinates generated from the defense tool, that is, (A1, B1, B2, A2, B3, B4, B5, ..., An, Bn). Thus, the attacker collects the X and Y coordinates on the screen by calling the GetCursorPos() function, and the time when the coordinates change. If the X and Y coordinates on the screen are the same as the previous and current coordinates, it is determined that there is no mouse input, and these coordinates are not collected. In other words, the coordinates are collected only when the previous coordinate and the current coordinate are different. The time when the coordinate is changed is stored as an elapsed time in nano seconds from the previous coordinate to the current coordinate, when the current coordinate is different from the previous coordinate. Table 1 shows an example of the collected coordinate.

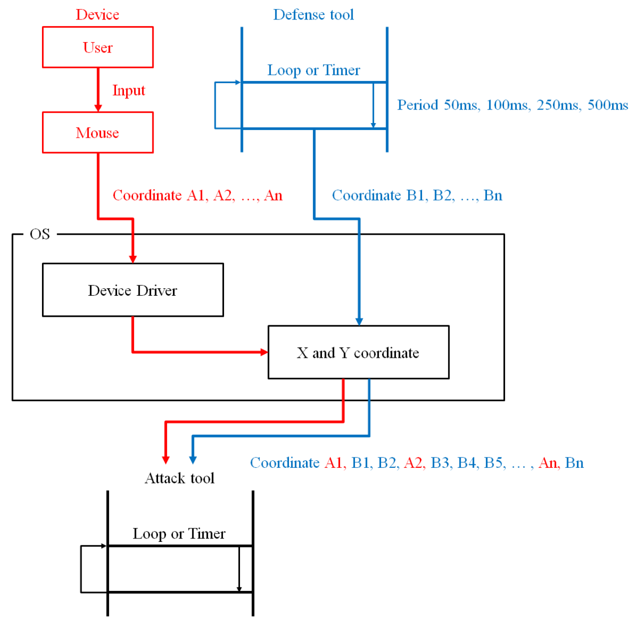


FIGURE 6. The proposed mouse data attack system configuration.

TABLE 1. Example of collected coordinates.

Index	elapsed time	X coordinate	Y coordinate	Generated coordinate	Real coordinate
29	1.69722784	0x2bd	0x19e	T	F
30	1.76087558	0x388	0xff	T	F
31	1.82509494	0x452	0x3a0	T	F
32	1.83707082	0x450	0x3a0	F	T
33	1.85703611	0x44e	0x3a2	F	T
34	1.87691462	0x44b	0x3a8	F	T
35	1.88418818	0x520	0x13d	T	F
36	1.90300238	0x519	0x148	F	T
…	…	…	…	…	…
25099	562.03942871	0x11e	0x22b	T	F
25100	562.09136963	0x11e	0x22c	F	T
25101	562.10363770	0x1ec	0x309	T	F

The goal of this paper is to classify the actual mouse data input from the mouse device, (A1, A2, …, An), from all the coordinates collected by the attack tool, namely, (A1, B1, B2, A2, B3, B4, B5, …, An, Bn). To achieve this goal, we utilize machine learning models of KNN, Logistic Regression, Linear SVC, Decision Tree, Random Forest, Gradient Boosting Regression Tree, SVM, and MLP. Moreover, we generated eight datasets for the experiment, and verified the feasibility of real mouse data exposure.

B. DATASET CONFIGURATION

To classify the actual mouse data coming from the mouse device, we collected mouse data in two ways. The first method collects both random data generated by the defense tool, and the actual mouse data input from the user. The second method collects only random mouse data. Each method in this paper collects data by setting four periods: of (50, 100, 250, and 500) ms. Table 2 shows the total collected mouse data.

TABLE 2. Whole collected mouse data.

Methods	Collected coordinates	Generation interval (ms)	Dataset
1	25,104	50	1-1
1	25,061	100	1-2
1	15,009	250	1-3
1	15,725	500	1-4
2	10,034	50	2-1
2	10,010	100	2-2
2	10,004	250	2-3
2	9,845	500	2-4

TABLE 3. Configured dataset for the experiment.

Dataset	Total collected coordinates	Number of benign coordinates	Percentage of benign coordinates (%)	Number of malignant coordinates	Percentage of malignant coordinates (%)
1-1	25,104	16,623	66.21	8,481	33.78
1-2	25,061	11,364	45.34	13,697	54.65
1-3	15,009	7,947	52.94	7,062	47.05
1-4	15,725	10,074	64.06	5,651	35.93
2-1	35,138	16,628	47.32	18,510	52.68
2-2	35,071	11,365	32.40	23,706	67.59
2-3	25,013	7,952	31.79	17,061	68.20
2-4	25,570	10,147	39.68	15,423	60.31

Datasets for the experiment ware constructed based on the collected mouse data. The coordinates collected by the second method are all coordinates generated by the defense tool, which are not organized into datasets for classification, but integrated with datasets (1-1 to 1-4) to improve the experiment results. In other words, datasets (1-1 to 1-4) are used for the experiment, and datasets (2-1 to 2-4) are integrated with datasets (1-1 to 1-4) for the experiment. Table 3 shows the dataset configuration for the experiment.

We constructed eight datasets for experiments, (1-1 to 1-4) and (2-1 to 2-4). Datasets (1-1 and 1-2) collected more than 25,000 coordinates, while (1-3 and 1-4) collected more than 15,000 coordinates. Benign coordinates in datasets represent actual coordinates input from the mouse device, while malignant coordinates represent random coordinates generated by a defense tool. The number of benign coordinates in each dataset is (16,623, 11,364, 7,947, 10,074, 16,628, 11,365, 7,952, and 10,147), while the number of malignant coordinates is (8,481, 13,697, 7,062, 5,651, 18,510, 23,706, 17,061, and 15,423), respectively. The percentages of benign coordinates are (66.21, 45.34, 52.94, 64.06, 47.32, 32.40, 31.79, and 39.68) %, while the percentages of malignant coordinates are (33.78, 54.65, 47.05, 35.93, 52.68, 67.59, 68.20, and 60.31) %. Therefore, in this paper, experiment datasets contain the appropriate number of benign and malignant coordinates, which reduce the problem caused by overfitting and underfitting.

For the experiment, features were organized in two ways. The first feature is the collected X and Y coordinates, while the second feature is the collected X and Y coordinate and the elapsed time. Therefore, the datasets used for the experiment were a total of 16 datasets using 8 datasets in two ways.

C. MACHINE LEARNING MODELS

This section describes the machine learning models used in this paper to classify mouse data, and these models are KNN, linear model, decision tree, decision tree ensemble, kernel trick, and neural network.

KNN is designated a class which has many neighbors as a label, and classified data based on decision boundaries [12]. Linear models use linear functions for classification. The linear classification models used in this paper are Logistic Regression and Linear Support Vector Machine (LinearSVC), which classify classes based on lines, planes, and hyperplanes [13], [14]. Decision Tree splits the data based on yes/no questions, and learns by repeating questions until a decision is reached [15]. Ensemble model is a model that creates an effective model based on several machine learning models [16], and includes Random Forest model and Gradient Boosting Regression model. Kernel trick includes Kernel Support Vector Machine (SVM), which determines decision boundaries based on points in each training data, and classifies by measuring the distance to the data point located at the boundary. Finally, deep learning model utilized multilayer perceptron (MLP) in this paper. MLP is a model that calculates the sum of weights by constructing hidden units in a linear regression model, and supports deep learning due to the hidden layer [18]. This model computes the sum of the weights of the hidden units to be more powerful than the linear model, and applies the nonlinear functions such as ReLU (refied linear unit) or hyperbolic tangent to the results. Moreover, this model is called deep learning because of the neural network composed of many hidden layers. This model has advantages that extracting the information contained in a large amount of data and generating a complex model due to the hidden layer. These advantages are also superior to other machine learning models. Nevertheless, this model takes a long time to learn and has a disadvantage of requiring data preprocessing.

IV. EXPERIMENT RESULT

This section describes the results of experiment and performance evaluations applied to machine learning models using datasets collected from the mouse data attack system described in Section 3. The machine learning models used in this paper are KNN, Logistic Regression, Linear SVC, Decision Tree, Random Forest, Gradient Boosting Regression Tree, SVM, and MLP.

A. EXPERIMENT RESULTS USING DATASETS CONSISTING OF FEATURE 1 (X AND Y COORDINATE ONLY)

For the experiment, the training set, the verification set, and the test set were classified into random numbers, and Table 4 shows the results of each set and the cross-validation results based on the dataset 1-1.

TABLE 4. **Training set, validation set, and test set scores in dataset 1-1 with optimal parameters.**

Model	Parameters	Training set score	Validation set score	Test set score	Cross-validation score
KNN	n_neighbors=29	0.67	0.66	0.65	0.656
Logistic regression	C=0.00001, L2 regularization	0.66	0.67	0.65	0.664
Linear SVC	-	0.66	0.67	0.65	0.664
Decision tree	max_depth=2	0.66	0.67	0.65	0.664
Random forest	n_estimators=13	0.98	0.57	0.54	0.559
Gradient boosting	learning_rate=1, max_depth=1	0.67	0.67	0.65	0.664
SVM	C=1	0.66	0.67	0.65	0.664
MLP	max_iter=1, alpha=0.00001	0.66	0.67	0.65	0.664

As an experiment result, random forest model has the highest score of 0.98 in the training set, while the rest of the models have a similar score of 0.66. The validation set has the lowest score at random forest with 0.57, while the other models have a similar score with 0.67. The test set has the lowest score at random forest with 0.54, while the other models have a similar score with 0.65. The cross-validation has the lowest score at random forest with 0.559, while the other models have a similar score with 0.66.

In order to evaluate the performance of the classification of the actual mouse data input from the mouse, the results of each set of cross-validation, accuracy, precision, recall, F1-score, and AUC were derived from datasets (1-1 to 1-4), as shown in Fig. 7. Cross-validation is a method of dividing the data over and over, and training multiple models. This cross-validation can improve the problem of performance increase or decrease by biasing the data. Accuracy is the exact predicted number (True Positive and True Negative, TP and TN) divided by the total number of samples, and precisely measures how many of the positively predicted (TP + False Positive, TP + FP) samples are truly positive (TP). Recall is measured by how many of the total positive samples (TP + False Negative, TP + FN) are classified as positive classes (TP). The F1-score is the harmonic mean of precision and recall, and is summarized in one. Area Under the Curve (AUC) summarizes the ROC curve with the area under the ROC curve, and the AUC result always has a value between the worst zero and the best one. The Receiver Operating Characteristics (ROC) curve considers all the thresholds of the classifier, and represents the False Positive Rate (FPR)

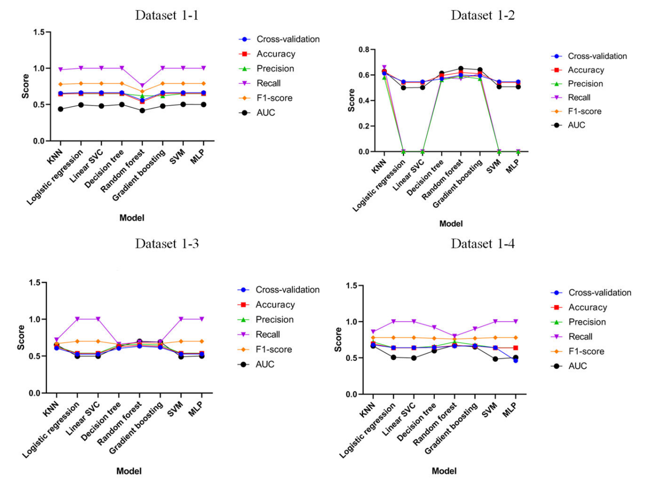


FIGURE 7. Performance evaluation results of datasets (1-1 to 1-4) (cross-validation, accuracy, precision, recall, F1-score, AUC).

against the True Positive Rate (TPR). The true positive rate is the recall rate, while the false positive rate is misclassified as false positive.

As a result of each dataset, dataset 1-1 shows high performance except for random forest, while dataset 1-2 shows high performance with KNN, decision tree, random forest, and gradient boosting regression tree. However, models that cannot measure recall are logistic regression, linear SVC, SVM, and MLP. Datasets 1-3 and 1-4 show high performance with KNN, decision tree, random forest, and gradient boosting regression tree, while logistic regression, linear SVC, SVM, and MLP have slightly lower performance. As a result of each model, all models have high performance in datasets (1-1 and 1-4). On the other hand, several models do not evaluate the performance in dataset 1-2, and these models are logistic regression, linear SVC, SVM, and MLP.

In order to evaluate the performance of the classification of the actual mouse data input from the mouse, the results of each set of cross-validation, accuracy, precision, recall, F1-score, and AUC were derived from datasets (2-1 to 2-4), as shown in Fig. 8:

As a result of each dataset, dataset 2-1 has the highest performance in the decision tree, while KNN, random forest, and gradient boosting regression tree have the highest performance. Dataset 2-2 has the highest performance with KNN and random forest, while dataset 2-3 has the highest performance with KNN, random forest, and gradient boosting regression tree. Dataset 2-4 has the highest performance with KNN, decision tree, random forest, and gradient boosting regression tree. As a result of each model, KNN, decision tree, random forest, and gradient boosting regression tree have the highest performance in dataset 2-4, while logistic regression, linear SVC, SVM, and MLP have the highest performance in datasets 2-2 and 2-3. A distinctive characteristic of these datasets is that models occur where precision, recall, and F1-score are not evaluated in all datasets.

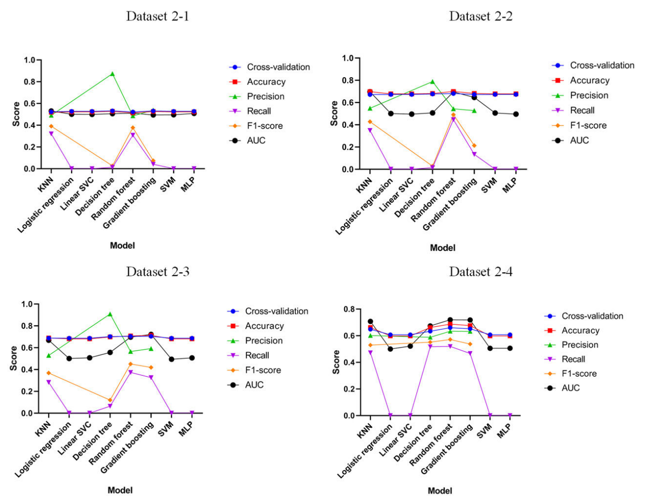


FIGURE 8. Performance evaluation results of datasets (2-1 to 2-4) (cross-validation, accuracy, precision, recall, F1-score, AUC).

TABLE 5. Training set, validation set, and test set scores in dataset 1-1 including elapsed time with optimal parameters.

Model	Parameters	Training set score	Validation set score	Test set score	Cross-validation
KNN	n_neighbors=30	0.75	0.75	0.75	0.750
Logistic regression	C=10, L2 regularization	0.75	0.76	0.75	0.751
Linear SVC	-	0.75	0.76	0.75	0.751
Decision tree	max_depth=4	0.75	0.76	0.75	0.750
Random forest	n_estimators=19	0.99	0.73	0.73	0.729
Gradient boosting	learning_rate=0.1, max_depth=2	0.76	0.76	0.75	0.753
SVM	C=100	0.75	0.76	0.75	0.752
MLP	max_iter=100, alpha=0.1	0.75	0.76	0.75	0.751

B. EXPERIMENT RESULTS USING DATASETS CONSISTING OF FEATURE 2 (X AND Y COORDINATE, AND ELAPSED TIME)

For the experiment, the training set, verification set, and test set were classified into random numbers, and Table 5 shows the results of each set and the cross-validation results based on dataset 1-1.

As an experiment result, the training set has the highest score at random forest with 0.99, while the other models have a similar score to 0.75. When compared to the dataset without elapsed time, random forest increased 0.01 from (0.98 to 0.99), while the other models increased 0.09 from (0.66 to 0.75). The validation set has the lowest score at random forest

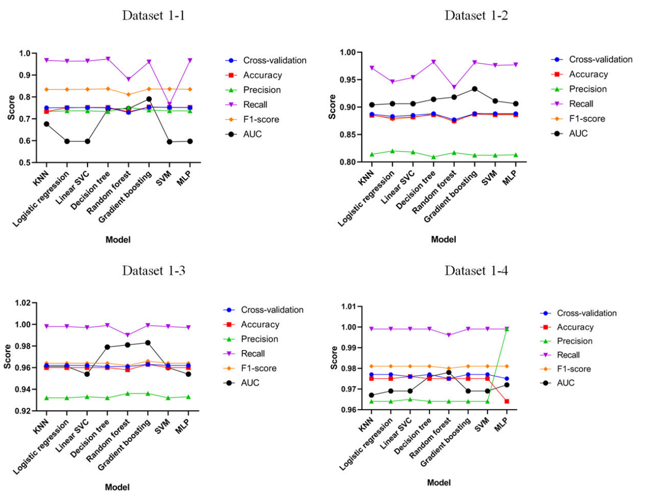


FIGURE 9. Performance evaluation results of datasets (1-1 to 1-4) with elapsed time (cross-validation, accuracy, precision, recall, f1-score, AUC).

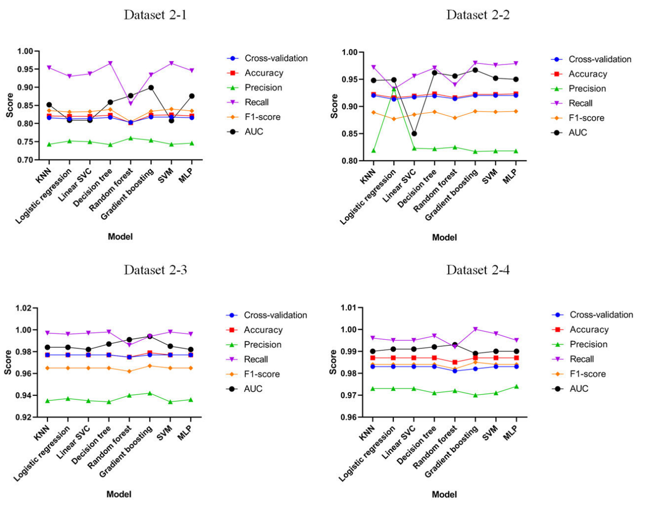


FIGURE 10. Performance evaluation results of datasets (2-1 to 2-4) with elapsed time (cross-validation, accuracy, precision, recall, f1-score, AUC).

with 0.73, while the other models have a similar score with 0.76. When compared to the dataset without elapsed time, random forest increased 0.16 from (0.57 to 0.73), while the other models increased 0.06 from (0.67 to 0.73). The test set has the lowest score at random forest with 0.73, while the other models have a similar score to 0.75. When compared to the dataset without elapsed time, random forest increased 0.19 from (0.54 to 0.73), while the other models increased 0.1 from (0.65 to 0.75). Cross-validation also has the lowest score at random forest with 0.729, while the other models have a similar score to 0.750. When compared to the dataset without elapsed time, random forest increased 0.170 from (0.559 to 0.729), while the other models increased 0.09 from (0.66 to 0.75). Thus, when compared to datasets that do not include elapsed time, the scores of all training sets, validation sets, test sets, and cross-validation scores are higher than those of the datasets that do not include elapsed time. This means including elapsed time results in high performance. In other words, if the dataset including elapsed time is used, the actual mouse data input from the mouse is classified more effectively. In order to analyze the performance evaluation results more precisely, the results of each set of cross-validation, accuracy, precision, recall, F1-score, and AUC were derived from datasets 1-1 to 1-4 as shown in Fig. 9.

As a result of each dataset, datasets 1-1 and 1-3 have the highest performance with decision tree, random forest, and gradient boosting regression tree, while dataset 1-3 has almost similar performance to all models, except random forest. Dataset 1-4 has the highest performance with decision tree and random forest. Most of all, all datasets, including elapsed time, have no model that does not evaluate performance in all models, when compared to datasets containing only X and Y coordinates. Namely, this means that the feature for distinguishing benign from malignant is properly defined. As a result of each model, all models have the highest performance in dataset 1-4.

In order to evaluate the classification performance of the actual mouse data input from the mouse, the results of each set of cross-validation, accuracy, precision, recall, F1-score, and AUC were derived from datasets (2-1 to 2-4) including elapsed time as shown in Fig. 10.

As a result of each dataset, dataset 2-1 has the highest performance with decision tree and gradient boosting regression tree, while dataset 2-2 has almost similar performance with all models, except linear SVC. Datasets 2-3 and 2-4 have almost similar performance with all models. As a result of each model, all models have the highest performance in dataset 2-4.

C. PERFORMANCE COMPARISON RESULTS ACCORDING TO FEATURES

As described above, in this paper, we evaluated the performance of datasets with feature 1, containing only X and Y coordinates, and datasets with feature 2, containing X and Y coordinates and elapsed time. As a result, we verified that the performance of the dataset including elapsed time is higher than that of the dataset containing only X and Y coordinates. Therefore, the results of the performance evaluation vary according to the feature, and the meaning of the feature is analyzed by comparing the performance evaluation. For this purpose, the results of the training set, verification set, test set, and cross-validation are compared, and the results of the actual performance of accuracy, precision, recall, F1-score, and AUC are compared. Fig. 11 shows the comparison results of the training set, verification set, test set, and cross-validation.

Based on the dotted line, the results of the datasets without elapsed time as shown on the left, while the results of the datasets with elapsed time as shown on the right. The performance tends to increase as the dataset goes from (1 to 3), and all datasets with elapsed time have higher performance than all datasets without elapsed time. Moreover, all scores

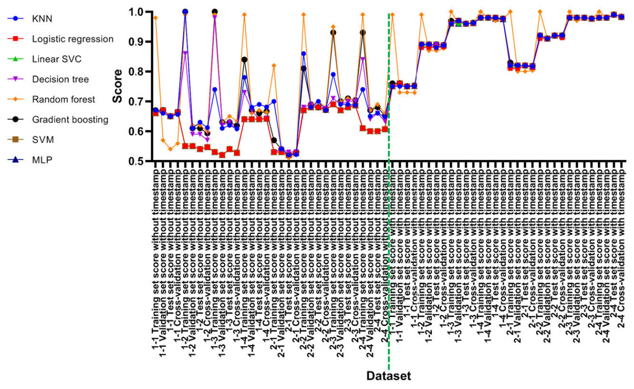


FIGURE 11. Performance evaluation comparison of the training set, validation set, test set, and cross-validation by features.

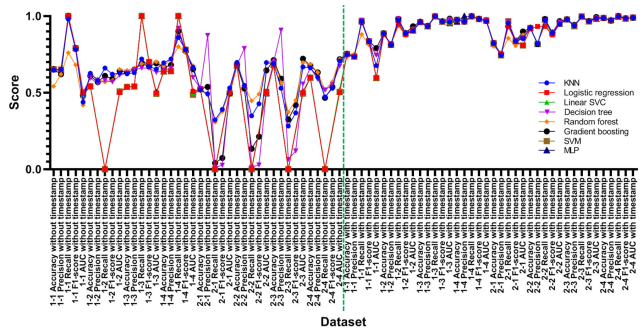


FIGURE 12. Performance evaluation comparison of accuracy, precision, recall, F1-score, and AUC by features.

of (1-4 and 2-4) have a performance close to the perfect score of 1, which results in a performance gain by constructing the feature into a dataset containing elapsed time.

By analyzing the changes in training set, validation set, test set, and cross-validation, the results of all models in the datasets that do not include elapsed time change significantly. In particular, decision tree, random forest, logistic regression, and gradient boosting regression tree show significant differences. On the other hand, the results of all models of the datasets containing elapsed time are relatively small. In these datasets, random forest shows noticeable difference.

As described above, the results of training set, validation set, test set, and cross-validation are clearly different according to the features, and the datasets including elapsed time show high performance. For more practical performance evaluation, Fig. 12 compares the results of accuracy, precision, recall, F1-score, and AUC.

Based on the dotted line, the results of the datasets without elapsed time are shown on the left, while the results of the datasets with elapsed time are shown on the right. As the datasets go from (1 to 4), the performance tends to improve, and datasets with elapsed time have relatively higher performance than datasets without elapsed time. Moreover, the accuracy of datasets (1-3, 1-4, 2-3, and 2-4) show a performance close to the perfect score of 1, which means that constructing the datasets with the feature of elapsed time

TABLE 6. Change rate according to the features in dataset 1-1.

Model	D	AC	+-	P	+-	R	+-	F	+-	AU	+-
KNN	w/o	0.646	115.9	0.65	**112.7**	0.98	98.6	0.78	106.9	0.438	154.3
	w	0.749		0.733		0.967		0.834		0.676	
Logistic regression	w/o	0.653	**114.8**	0.65	113.2	1	96.3	0.79	**105.5**	0.496	120.3
	w	0.75		0.736		0.963		0.834		0.597	
Linear SVC	w/o	0.653	115.1	0.65	113.2	1	96.4	0.79	105.6	0.481	124.1
	w	0.752		0.736		0.964		0.835		0.597	
Decision tree	w/o	0.653	115.1	0.65	**112.7**	1	97.3	0.79	105.9	0.501	148.7
	w	0.752		0.733		0.973		0.837		0.745	
Random forest	w/o	0.541	**135.3**	0.62	**121.2**	0.76	**115.7**	0.68	**119.2**	0.418	**178.4**
	w	0.732		0.752		0.88		0.811		0.746	
Gradient boosting	w/o	0.653	115.4	0.62	119.3	1	**96**	0.79	105.8	0.479	164.9
	w	0.754		0.74		0.96		0.836		0.79	
SVM	w/o	0.653	115.1	0.65	113.2	1	96.6	0.79	105.8	0.503	**118.2**
	w	0.752		0.736		0.966		0.836		0.595	
MLP	w/o	0.653	115.1	0.65	113.2	1	96.6	0.79	105.6	0.5	119.4
	w	0.752		0.736		0.966		0.835		0.597	

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

results in higher performance. Therefore, the accuracy of nearly 1 extracts most of the actual mouse data input from the user by effectively classifying random coordinates generated by the defense tool in image-based authentication.

By analyzing the changes in accuracy, precision, recall, F1-score, and AUC, the results of all models in datasets without elapsed time change significantly. In particular, logistic regression, linear SVC, decision tree, random forest, and gradient boosting regression tree show significant differences. On the other hand, the results of all modes in datasets with elapsed time are relatively small. In these datasets, random forest and logistic regression show noticeable differences.

Finally, in order to analyze the increase and decrease of the change rate according to features, accuracy, precision, recall, F1-score, and AUC performance, the difference of datasets having the lowest performance and datasets having the highest performance are analyzed. Tables 6-9 show the results.

The results in dataset 1-1 show that the model with the highest increase of accuracy was random forest, which increased by 135.3 %; while the model with the lowest increase of accuracy was logistic regression, which increased by 114.8 %. The model with the highest increase of precision was random forest, which increased by 121.2 %; while the models with the lowest increase of precision were KNN and decision tree, which increased by 112.7 %. The model with the highest increase of recall was random forest, which increased by 115.7 %; while recall showed that gradient boosting regression tree had a lower evaluation of datasets including elapsed time, and decreased to 96 %. The model with the highest increase of F1-score was random forest, which increased by 119.2 %; while the model with the lowest increase of F1-score was logistic regression, which increased by 105.5 %. The model with the highest increase of AUC was random forest, which increased by 178.4 %; while the model with the lowest increase of AUC was SVM, which

TABLE 7. Change rate according to the features in dataset 1-3.

Model	D	AC	+-	P	+-	R	+-	F	+-	AU	+-
KNN	w/o	0.623	154.0	0.63	147.9	0.72	138.6	0.67	143.8	0.654	146.9
	w	0.96		0.932		0.998		0.964		0.961	
Logistic regression	w/o	0.537	178.7	0.54	172.5	1	99.8	0.7	137.7	0.499	192.5
	w	0.96		0.932		0.998		0.964		0.961	
Linear SVC	w/o	0.537	178.7	0.54	172.7	1	99.7	0.7	137.7	0.499	191.1
	w	0.96		0.933		0.997		0.964		0.954	
Decision tree	w/o	0.629	152.6	0.65	143.3	0.66	151.3	0.66	146.0	0.632	154.9
	w	0.96		0.932		0.999		0.964		0.979	
Random forest	w/o	0.645	148.5	0.66	141.8	0.69	143.4	0.68	141.4	0.701	139.9
	w	0.958		0.936		0.99		0.962		0.981	
Gradient boosting	w/o	0.632	152.3	0.65	144	0.69	144.7	0.67	144.1	0.692	142.0
	w	0.963		0.936		0.999		0.966		0.983	
SVM	w/o	0.537	178.7	0.54	172.5	1	99.8	0.7	137.7	0.492	195.1
	w	0.96		0.932		0.998		0.964		0.96	
MLP	w/o	0.537	178.7	0.54	172.7	1	99.7	0.7	137.7	0.502	190.0
	w	0.96		0.933		0.997		0.964		0.954	

TABLE 8. Change rate according to the features in dataset 2-2.

Model	D	AC	+-	P	+-	R	+-	F	+-	AU	+-
KNN	w/o	0.697	132.2	0.548	149.4	0.349	278.5	0.427	208.1	0.696	136.2
	w	0.922		0.819		0.972		0.889		0.948	
Logistic regression	w/o	0.677	135.3	-	-	0	-	-	-	0.5	189.8
	w	0.916		0.932		0.932		0.877		0.949	
Linear SVC	w/o	0.677	135.7	-	-	0	-	-	-	0.495	191.9
	w	0.919		0.823		0.956		0.885		0.95	
Decision tree	w/o	0.68	135.7	0.788	104.3	0.014	6935.7	0.028	3178.5	0.506	190.1
	w	0.923		0.822		0.971		0.89		0.962	
Random forest	w/o	0.699	131.0	0.543	151.9	0.446	210.7	0.49	179.3	0.695	137.5
	w	0.916		0.825		0.94		0.879		0.956	
Gradient boosting	w/o	0.681	135.3	0.527	155.0	0.134	731.3	0.213	418.3	0.645	149.9
	w	0.922		0.817		0.98		0.891		0.967	
SVM	w/o	0.677	136.1	-	-	0	-	-	-	0.505	188.5
	w	0.922		0.818		0.976		0.89		0.952	
MLP	w/o	0.677	136.3	-	-	0	-	-	-	0.495	191.9
	w	0.923		0.818		0.979		0.891		0.95	

TABLE 9. Change rate according to the features in dataset 2-4.

Model	D	AC	+-	P	+-	R	+-	F	+-	AU	+-
KNN	w/o	0.662	149.0	0.601	161.8	0.472	211.0	0.529	186.0	0.707	140.0
	w	0.987		0.973		0.996		0.984		0.99	
Logistic regression	w/o	0.598	165.0	-	-	0	-	-	-	0.5	198.2
	w	0.987		0.973		0.995		0.984		0.991	
Linear SVC	w/o	0.598	165.0	-	-	-	-	-	-	0.521	190.2
	w	0.987		0.973		0.995		0.984		0.991	
Decision tree	w/o	0.661	149.3	0.588	165.1	0.517	192.8	0.551	178.5	0.673	147.3
	w	0.987		0.971		0.997		0.984		0.992	
Random forest	w/o	0.686	143.5	0.633	153.5	0.519	191.1	0.571	171.9	0.719	138.1
	w	0.985		0.972		0.992		0.982		0.993	
Gradient boosting	w/o	0.676	146.0	0.632	153.4	0.466	214.5	0.537	183.4	0.718	137.7
	w	0.987		0.97		1		0.985		0.989	
SVM	w/o	0.598	165.0	-	-	-	-	-	-	0.505	196.0
	w	0.987		0.971		0.998		0.984		0.99	
MLP	w/o	0.598	165.0	-	-	-	-	-	-	0.506	195.6
	w	0.987		0.974		0.995		0.984		0.99	

increased by 118.2 %. In dataset 1-1, all evaluation results were increased, except for gradient boosting regression tree in recall, with the highest growth rate of 178.4 %.

The results in dataset 1-3 show that the models with the highest increase of accuracy were logistic regression, linear SVC, SVM, and MLP, which increased by 178.7 %; while the model with the lowest increase of accuracy was random forest, which increased by 148.5 %. The models with highest increase of precision were linear SVC and MLP, which increased by 172.7 %; while the model with the lowest increase of precision was random forest, which increased by 141.8 %. The model with the highest increase of recall was decision tree, which increased by 151.3 %; while recall showed that linear SVC and MLP had a lower evaluation of datasets including elapsed time, and decreased to 99.7 %. The model with the highest increase of F1-score was decision tree, which increased by 146 %; while the models with the lowest increase of f1-score were logistic regression, linear SVC, SVM, and MLP, which increased by 137.7 %. The model with the highest increase of AUC was SVM, which increased by 195.1 %; while the model with the lowest increase of AUC was random forest, which increased by 139.9 %. In dataset 1-3, all evaluation results were increased except for linear SVC and MLP in recall, with the highest growth rate of 195.1 %.

The results in dataset 2-2 show that the model with the highest increase of accuracy was MLP, which increased by 136.3 %; while the model with the lowest increase of accuracy was random forest, which increased by 131 %. The model with the highest increase of precision was gradient boosting, which increased by 155 %; while the model with the lowest increase of precision was decision tree, which increased by 104.3 %. The model with the highest increase of recall was decision tree, which increased by 6,935.7 %; while the model with the lowest increase of recall was random forest, which increased by 210.7 %. The model with the highest increase of f1-score was decision tree, which increased by 3,178.5 %; while the model with the lowest increase of f1-score was random forest, which increased by 179.3 %. The model with the highest increase of AUC was linear SVC, which increased by 191.9 %; while the model with the lowest increase of AUC was KNN, which increased by 136.2 %. Dataset 2-2 showed an increase in the evaluation results of all models, which the highest increase rate of 6,935.7%.

The results in dataset 2-4 show that the models with the highest increase of accuracy were logistic regression, linear SVC, SVM, and MLP, which increased by 165 %; while the model with the lowest increase of accuracy was random forest, which increased by 143.5 %. The model with the highest increase of precision was decision tree, which increased by 165.1 %; while the model with the lowest increase of precision was gradient boosting regression tree, which increased by 153.4 %. The model with the highest increase of recall was gradient boosting regression tree, which increased by 214.5 %; while the model with the lowest increase of recall was random forest, which increased by 191.1 %. The model with the highest increase of f1-score was KNN which increased by 186 %; while the model with the lowest increase of f1-score was random forest, which increased by 171.9 %. The model with the highest increase of AUC was logistic regression, which increased by 198.2 %; while the model with the lowest increase of AUC was gradient boosting regression tree, which increased by 137.7 %. Dataset 2-4 showed an increase in the evaluation results of all models, with the highest increase rate of 214.5 %.

In conclusion, using datasets that includes elapsed time, we extracted the actual mouse data input from the user by effectively classifying random coordinates with up to 98 %

accuracy. Therefore, the attack method proposed in this paper has found a new vulnerability and security threat to effectively steal the user password in image-based authentication.

V. CONCLUSION

This paper analyzed the security of mouse data in the situation where defense technique is applied based on machine learning in image-based authentication. The existing attack technique is limited in its ability to distinguish the mouse data generated by the defense tool from the actual mouse data input from the user. In order to overcome this limitation, in this paper, we constructed datasets with features to effectively classify mouse data. As a result of experiments based on the configured datasets, the datasets collected in this paper classified the actual mouse data input from the user at a much higher level than did the existing attack technique. This means that the proposed system steals mouse data effectively. In particular, performance indicators, such as accuracy, precision, recall, F1-score, and AUC, were evaluated as higher than the existing attack in all datasets, and showed very low false positive and false negative rates. Moreover, the best accuracy is 98 %, which means that the attacker almost completely steals the mouse data. Therefore, the proposed method found a new vulnerability and security threat for image-based authentication technology. Finally, the results of this paper will be used as criteria for security analysis and evaluation in image-based authentication technology.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] A. Conklin, G. Dietrich, and D. Walz, "Password-based authentication: A system perspective," in *Proc. 37th Annu. Hawaii Int. Conf. Syst. Sci.*, Big Island, HI, USA, Jan. 2004, p. 10.

[2] K. Lee and K. Yim, "Password sniff by forcing the keyboard to replay scan codes," in *Proc. JWIS*, Guangzhou, China, Aug. 2010, pp. 9–11.

[3] K. Lee and K. Yim, "Keyboard security: A technological review," in *Proc. IMIS*, Seoul, South Korea, Jun./Jul. 2011, pp. 9–15.

[4] I. Oh, K. Lee, S.-Y. Lee, K. Do, H. B. Ahn, and K. Yim, "Vulnerability analysis on the image-based authentication through the PS/2 interface," in *Proc. IMIS*. Matsue, Japan: Shimane Prefectural Convention Center, Jul. 2018, pp. 212–219.

[5] T. Takada and H. Koike, "Awase-E: Image-based authentication for mobile phones using user's favorite images," in *Proc. Mobile HCI*. Berlin, Germany: Springer, Sep. 2003, pp. 347–351.

[6] A. Parekh, A. Pawar, P. Munot, and P. Mantri, "Secure authentication using anti-screenshot virtual keyboard," *Int. J. Comput. Sci. Issues*, vol. 8, no. 5, pp. 534–537, Sep. 2011.

[7] R. E. Newman, P. Harsh, and P. Jayaraman, "Security analysis of and proposal for image-based authentication," in *Proc. 39th Annu. Int. Carnahan Conf. Secur. Technol.*, Las Palmas, Spain, Oct. 2005, pp. 141–144.

[8] H. Lee, Y. Lee, K. Lee, and K. Yim, "Security assessment on the mouse data using mouse loggers," in *Proc. BWCCA*. Asan, South Korea: Soonchunhyang Univ., Oct. 2016, pp. 387–393.

[9] Q. Do, B. Martini, and K.-K. R. Choo, "The role of the adversary model in applied security research," *Comput. Secur.*, vol. 81, pp. 156–181, Mar. 2019.

[10] Q. Do, B. Martini, and K.-K. R. Choo, "Cyber-physical systems information gathering: A smart home case study," *Comput. Netw.*, vol. 138, pp. 1–12, Jun. 2018.

[11] I. Oh, K. Lee, and K. Yim, "A protection technique for screen image-based authentication protocols utilizing the SetCursorPos function," in *Proc. WISA*, Jeju Island, South Korea, Aug. 2017, pp. 236–245.

[12] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *J. Pattern Recognit.*, vol. 40, no. 7, pp. 2038–2048, Jul. 2007.

[13] W. Cheng and E. Hüllermeier, "Combining instance-based learning and logistic regression for multilabel classification," *Mach. Learn.*, vol. 76, nos. 2–3, pp. 211–225, Sep. 2009.

[14] B. Schölkopf and A. J. Smola, *Learning With Kernels*. Cambridge, MA, USA: MIT Press, 2002.

[15] C. Sinclair, L. Pierce, and S. Matzner, "An application of machine learning to network intrusion detection," in *Proc. ACSAC*, Washington, DC, USA, Dec. 1999, pp. 371–377.

[16] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "A comparison of decision tree ensemble creation techniques," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 173–180, Jan. 2007.

[17] S. U. Jan, Y. D. Lee, J. Shin, and I. Koo, "Sensor fault classification based on support vector machine and statistical time-domain features," *IEEE Access*, vol. 5, pp. 8682–8690, May 2017.

[18] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.