Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

VR-Sniffer: WebVR Browsing History Sniffing Attack by Exploiting Controller Input Leakage

JIYEON LEE[1], BYUNGGILL JOE[2], AND KILHO LEE.[1], (Member, IEEE)
[1]School of AI Convergence, Soongsil University, Seoul 06978, Republic of Korea (e-mail: jylee.cs@ssu.ac.kr, khlee.cs@ssu.ac.kr)
[2]School of Computing, KAIST, Daejeon 34141, Republic of Korea (e-mail: cp4419@kaist.ac.kr)

Corresponding author: Kilho Lee (e-mail: khlee.cs@ssu.ac.kr).

This work was supported in part by MSIT, Korea, under the Innovative Human Resource Development for Local Intellectualization support program (IITP-2022-RS-2022-00156360) and National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2021R1A4A1032252).

ABSTRACT Recently, with a rapidly increasing interest in Metaverse, Virtual Reality (VR) comes to the web and widely spreads to various apps. This is enabled by the WebVR framework. It allows running VR apps on top of major web browsers, providing better compatibility and accessibility. However, WebVR is exposed to privacy threats due to the open nature of the web. In particular, a malicious website can easily obtain sensor data from a user's VR controller and distinguish the user's browsing history through the controller usage patterns. This paper presents such a threat and demonstrates an effective privacy attack on major web browsers. We propose VR-Sniffer, a deep learning-based browsing history identification platform. For privacy attacks, it collects the sensor data from the user's VR controller, transforms the data into a suitable format for our deep learning network, feeds the processed data into the network, and then infers the WebVR site that the user is visiting. Our extensive evaluation with 18 participants shows that VR-Sniffer effectively identifies the visiting WebVR site with an accuracy of up to 95.8%. This result implies that the proposed attack may result in severe privacy threats to major WebVR applications.

INDEX TERMS Web security, Privacy threats, WebVR, Browsing history sniffing, Side-channel attacks, Virtual reality, VR sensors, Controller usage pattern, Deep learning, Image classification

I. INTRODUCTION

In 2017, an open VR platform called WebVR [1] was introduced to facilitate the development and distribution of VR content. WebVR provides a VR app in the form of a website, allowing users to play it with browser support regardless of what VR device they have. With the advent of WebVR, VR developers can easily create VR content running on multiple devices in a web development language, and users can conveniently consume new VR content by accessing the website. With these strong advantages, WebVR is receiving keen attention from major browsers, and their showcases are proving the potential of WebVR [19].

Despite its numerous advantages, WebVR also poses privacy risks. Unlike native VR, WebVR, which has the characteristics of the web, can be opened along with multiple websites through multiple browser windows. The problem is that websites, including non-WebVR sites, can access sensor data on VR controllers provided by browsers without any permission, which means multiple sites can access it at the same time. This means that if a user visits and plays a benign WebVR site, the controllers' sensor data generated by the user's hand motion can be exposed to untrusted parties. This allows attackers to intercept the real-time pose and orientation of VR controllers as well as button-click events, leading to new privacy threats.

In this study, we introduce a VR sniffing attack that uses a security hole in WebVR to identify the WebVR site that victims are browsing. Identifying browsing history is a traditional web attack, which is considered a very threatening attack as it can leak personal information that users do not want to disclose, such as political orientation and dating preferences [2], [6]. Recently, various applications such as immersive news, 360-degree video players, and adult content are provided through VR, so it is an interesting topic to study whether it is possible to disclose the browsing history of

WebVR. WebVRs with UI buttons that must be clicked to proceed with the app create a unique input pattern, leaving room for identifying browsing history with controller sensor information exposed to the untrusted party.

Existing history sniffing methods proposed to identify non-WebVR sites have relied mainly on side-channel leaks such as timing information. In contrast, this paper proposes a novel attack method using the leaked information on VR controllers, where the following challenges exist: First, many VR devices support a standing mode, not a room-scale mode, in which the user's hand motion is relatively fixed, as a result, sensor data on VR controllers generated by playing each WebVR does not show a clear difference. In addition, when playing a WebVR, the user interaction sequence is not always the same. Controller inputs are affected by users' play style; users can selectively play on given UIs. Such randomness makes it difficult to specify the exact WebVR being played. It indicates that rule-based approaches are not suitable as the number of target WebVR sites increases.

To address the above challenges, we propose VR-Sniffer, a browsing history identification framework on WebVR. VR-Sniffer provides a WebVR site that runs with the target site and collects hit points and directions where the controller's laser point intersects the object we define in the VR scene for each click event. This additional information makes it easier to identify WebVR sites than the sensor data of the controller object itself. We also noticed that each WebVR displays a different graph shape when plotting the collected hit points. Based on this observation, we designed a deep-learning model with graph images as input. We have created two types of graph images with data augmentation that capture app-specific input features for model training. The classification model trained with our dataset shows stable performance regardless of the user's input sequence.

To evaluate VR-Sniffer, we have retrieved 12 real-world WebVR sites that use VR controllers on the Internet and collected 72 test data by recruiting 18 student participants. VR-Sniffer was able to identify the target WebVR with 95.8% accuracy when using 768 augmented data, which is a 55.4% improvement in performance over the baseline. In the TopK experiment to determine the classification accuracy for the kth attempt, VR-Sniffer showed 100% classification accuracy on the second attempt. VR-Sniffer has also been shown to cause degradation of target WebVR sites by an average page loading time of 31 msec and frame/second (FPS) of 0.64, indicating that the attack is possible with negligible impact on the performance of target sites.

A. RELATED WORK

History sniffing attacks that identify web browsing history, including currently open websites, have been considered a serious privacy threat. To the best of our knowledge, no related research has been conducted on WebVR. Stealing sensitive information in traditional web applications has been extensively studied [2]–[9]. It has been shown that differences in timing information (e.g., page loading time) of users' browsers can expose browsing histories [2], [7]. S. Chen et al. revealed that the size of transmitted packets can be used to infer highly sensitive information, such as web search queries, even when the presence of HTTPS protection [5]. Other studies have introduced new attack methods that exploit new features of the web [7]–[9]. J. Lee et al. introduced a new history sniffing attack abusing an offline browsing mode on progressive web apps [9]. They showed that history sniffing is possible with high accuracy compared to previous attacks that rely on side-channel leakage. Unlike previous studies, this paper presents a novel history sniffing attack that identifies visiting WebVR sites through VR controller sensor data leaked to untrusted parties.

B. CONTRIBUTION

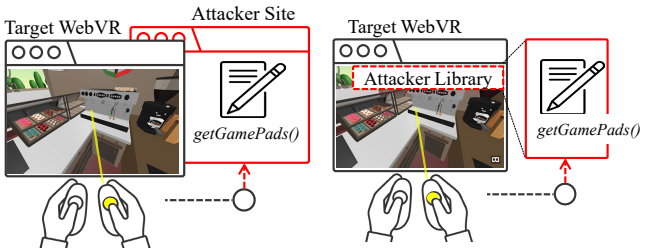
- For the first time, we present a VR sniffing attack that identifies the WebVR site that the victim is visiting. This attack is a novel side-channel attack exploiting leak information from VR controllers and targets real-world WebVR sites.
- We provide VR-Sniffer for WebVR identification by converting hit points into graph images. VR-Sniffer captures app-specific input patterns well while being less affected by the user's play style.
- Our extensive experiments show that VR-Sniffer is able to identify target WebVRs with very high accuracy (an average of 95.8%). It also has little effect on the performance of target WebVRs during the attack, showing that it is a highly practical attack.

II. BACKGROUND

A. WEBVR

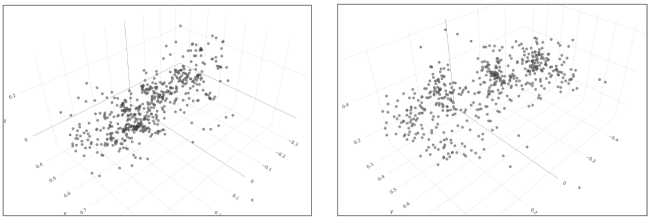
WebVR [1] is a new HTML5 standard technology that enables to running of VR by means of browser support. It offers the powerful advantage of the ease of developing and sharing VR content. Along with WebGL [10], WebVR enables an immersive 3D world experience by providing a set of VR-enabled interfaces that manage VR devices. A GamePad API [11] allows reading the pose and orientation of VR controllers on WebVR. It also provides a mapping table for controller buttons on various VR devices so that WebVR can track user click input. Due to its sensitivity, WebVR can only be executed in secure contexts (i.e., HTTPS) [11]. For simplicity, we refer to a WebVR site as a WebVR or a VR app throughout the remainder of the paper.

Recently, WebVR is being integrated into WebXR [12] that embraces various reality technologies such as Augmented Reality (AR) and Mixed Reality (MR). The most different underneath technology from WebVR is that WebXR manages an xrSession [13], which tracks the entire lifecycle of the XR app, providing an interface that can easily scale to a variety of "reality" environments. WebXR is currently under development and at the time of writing, it is partially supported by Chrome and Edge browsers [14]. WebVR, on



(a) WebVR Attacker (b) iframe Attacker

FIGURE 1: Attacker Model



(a) A-Blast (b) Moon Rider

FIGURE 2: Controller coordinates collected from two different VR apps

the other hand, is available as default in Firefox which is the only specification fully usable in a desktop environment [15]. In this paper, we focus on new security threats related to WebVR.

B. SECURITY RISKS ON WEBVR

Despite the WebVR API being maintained in secure contexts, we found an unexpected situation when using VR controllers with multiple browser windows. It turns out that the GamePad objects described above are running in the browser context without any protection, and as a result, they can be accessed simultaneously from multiple websites with cross-origins. This entails a critical security flaw that allows an untrusted website to gather all controller inputs generated for a benign WebVR while both websites are open in the same browser. Through this, an attacker can know the real-time pose and orientation as well as the time of clicking a button on VR controllers manipulated by the victim. The Gamepad object also records information such as hand direction, which is not covered in detail because of less related to our approach. The full specification of the GamePad object can be found in [11].

III. THREAT MODEL

Figure 1 shows two attack models for identifying browsing history on WebVR. We first define a WebVR attacker (Figure 1a), who serves his/her own WebVR site and can induce victims to visit this site. We assume that the attacker's WebVR is open to the same browser where the target WebVR is open during the attack. The proposed attack can also be carried out by a third-party JavaScript (JS) library attacker (Figure 1b). In this scenario, the target WebVR embeds the attacker's JS library for some functional purpose, which in fact, this JS code loads the malicious WebVR into an invisible iframe element [16].

We note that an attack site can be a non-WebVR website (i.e., 2D websites) since there are no restrictions on accessing GamePad objects with regular websites. Despite this, our attack method relies on a raycaster system [17] that can be defined in VR scenes, we insist on performing the attack over a WebVR (See Section V in more detail). Unlike traditional websites, WebVR requires VR permission [18]; it can only be run once the user has been granted permission. Nevertheless, the attacker's WebVR can acquire GamePad objects without any permission. Furthermore, there is no need to modify the target webVR or install malicious apps on the victim's computer.

IV. TARGET WEBVR SITES

For the target WebVRs to be identified, we referred to a WebVR directory site [19] managed by Supermedium [20], one of the major WebVR services. Compared to 2D websites, WebVR is a recently introduced technology, so there are not many sites yet; the WebVR directory lists as many as possible WebVRs developed by well-known developer communities such as Google Creative Lab [21] and Mozilla Labs [22], as well as individual developers. We collected a total of 46 WebVR sites from this reference site, excluding those currently not working (closed or non-executable) and those provided at the same URL.

Our attack exploits leak information from VR controllers, so the use of VR controllers is a minimum requirement. Therefore, we excluded VR apps that simply use head-mounted displays (e.g., VR apps for viewing purposes) from the target group. We also excluded VR apps such as Drawing apps and Minecraft apps that receive VR controller inputs completely randomly because our approach is required at least one static user interaction. We confirmed that 22 of the total collected WebVRs used VR controllers and 10 of them had no static UI at all. We classify them into an unclassified group and conduct experiments with 12 WebVRs that correspond to our attack. It is worth mentioning that there have been no studies dealing with such many WebVR sites, and we believe that these target WebVRs are sufficient to represent the categories of real-world VR apps. A detailed description of target WebVRs is provided in Table 1.

V. VR-Sniffer

The goal of this study is to infer the WebVR that the victim is currently playing through the controller input pattern induced by the VR app. We first describe the difficulties of WebVR identification only with the information provided by GamePad objects in Section V-A and introduce a solution in Section V-B.

TABLE 1: Description of Target VR Apps

App Name	Category	Description	Remarks
A-Blast	Game	A VR game that uses two laser guns to shoot down monsters in the sky.	A user must click the start button to continue. Shooting targets appear randomly within 120 degrees of the front.
Barista Express	Game	A VR game to become a barista and run a cafe.	Customers and coffee machines, where the user interacts actively, are mainly placed in the right and front directions.
Cross the Street	Game	A VR game in which one gains points by crossing a crosswalk while avoiding cars.	A user can walk or jump by quickly moving controllers up and down while pressing the button.
Jelly Face	Game & Utility	A VR app that allows users to pinch or squeeze celebrities' 3D faces.	There are UI buttons that change celebrities, which can be clicked at any time during play. The 3D face position is calculated as the HMD position (i.e., not fixed).
Access Mars	Education & Information	A VR app that explores Mars by looking around photographs provided by the Curiosity rover.	A user arbitrarily clicks on navigation buttons to browse the description or move to another place.
Moon Rider	Game & Fitness	A VR game that beats the rhythm along a road.	A user must select a song to play from the front panel for the start. Each song has a different rhythm.
A Saturday Night	Game & Utility	A VR app that records and shares a user's movements when they dance to music.	A user must select an avatar located in the front for the start. No button clicks are required while dancing.
Sound Boxing	Game & Fitness	A VR kickboxing game with beats.	A user must select a song to play from the front panel for the start. No button clicks are required while playing.
Space Rocks	Game	A VR game that shoots down asteroids while flying through space.	A user must click the start button to continue. Shooting targets appear randomly from all sides and vary in size, in which the clicking interval depends on the size of the object.
Trajectile Command	Game	A VR game to protect the city from missiles and bombers.	A user must click the start button to continue. The game is played in the cube space.
Witch's Brew	Utility	A VR app that mixes red, blue, and yellow potions to create a mysterious stew.	Potions and spatula are placed in the right and left directions. The user clicks on the controller buttons to grab them and pour or stir them into the jar.
Within	Utility	A 360-degree VR Video Player.	A user selects one of the videos displayed on the front to watch.

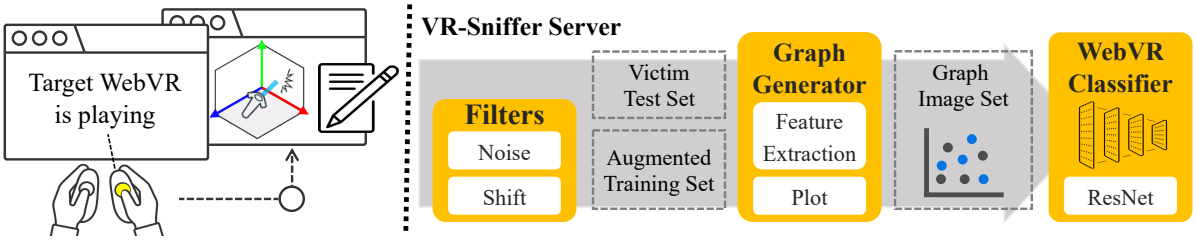


FIGURE 3: The overall process of VR-Sniffer

A. CHALLENGE

As mentioned in Section II, WebVRs share a GamePad object with the attacker's website. However, the information that the attacker can acquire through this object is raw data limited to the controller, and no information about the target WebVR (e.g., the type of app, and the scene being rendered) is known. In order to identify the WebVR with this limited information, the following problems should be considered.

1) The GamePad object stores the coordinates of the controller in the 3D space. However, coordinates generated by VR apps playing in standing mode are typically observed in limited space. Figure 2 shows the 3D coordinates of the controller collected for every click event in two different apps, which is not enough to identify the app. Therefore, VR-Sniffer should be able to extract more identifiable features from each VR app.
2) Even in a single VR app, user input sequences are not always the same. The user can selectively play on given UIs, and it is impossible to navigate all potential input sequences. Therefore, VR-Sniffer should be attackable regardless of how the user plays.

B. SOLUTION

Figure 3 shows the overall process of VR-Sniffer. VR-Sniffer first collects controller data from the target WebVR using the attacker's WebVR site. VR-Sniffer then sends the data to the server and preprocesses the data making WebVR identifiable. To address the first challenge, we introduce a technique that allows us to distinguish app-specific features well using a raycaster system provided by VR frameworks. For the second challenge, we convert the collected data into graph images and train this data with an image classification model. Through this, it is possible to understand the pattern of the entire input without focusing on the individual input, which is less influenced by the victim's play.

1) Data Collection

The attacker's website running along with the target WebVR is implemented as a WebVR. This WebVR defines

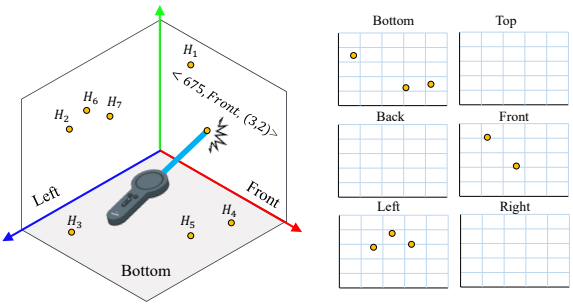


FIGURE 4: Illustration showing the process of collecting hit points

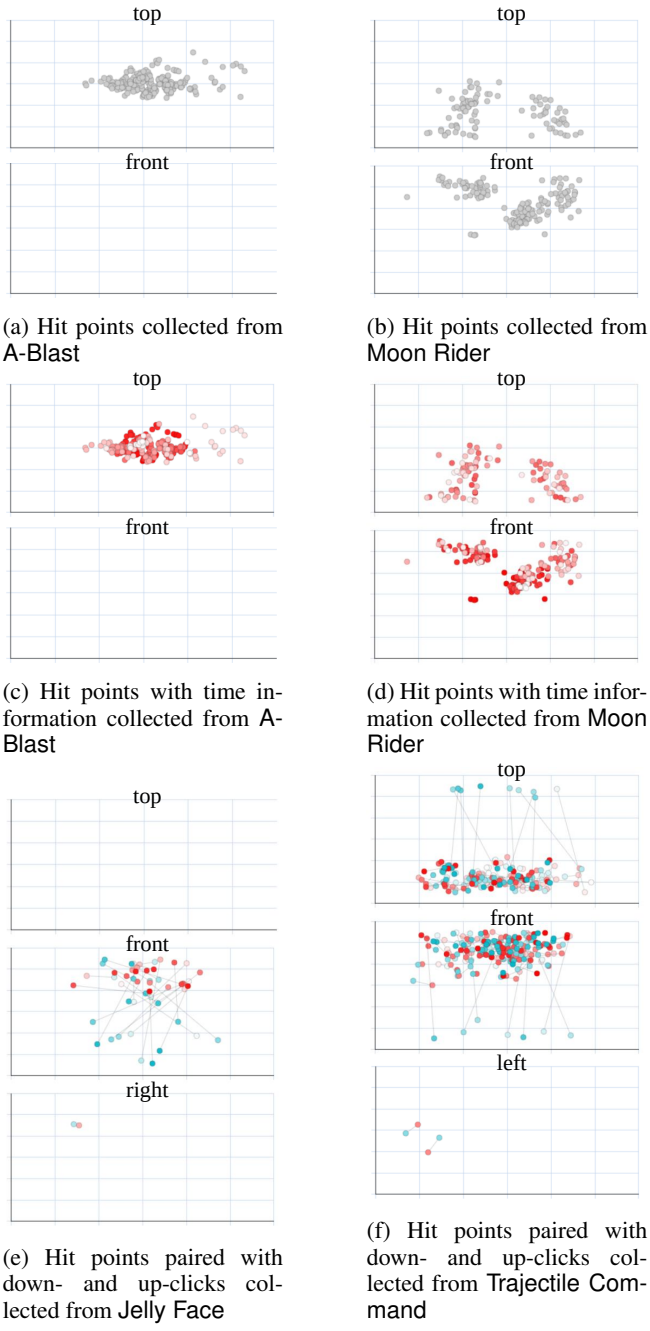
two hand controller objects presenting laser pointers, and defines down (press) and up-click (release) event handlers. VR-Sniffer records data only when a click event fires, which is reasonable considering that most user interactions using controllers are done with clicks. In addition, we render a cube object in the VR scene (center is $(0, 0, 0)$) and record a hit point, $\mathbf{H}_i \in \mathbb{R}^2$, where the laser point of the controller and one of the cube planes collide when the controller button is clicked (see Figure 4). The hit point is easily obtained by raycaster provided by most WebVR frameworks [23], [24], which are essential for WebVR development. In the end, the VR-Sniffer records $< t_i, d_i, \mathbf{H}_i >$ for ith click, where t_i denotes the clicked time, and d_i denotes the direction of the hitting plane among a hexahedron. VR-Sniffer sends all records to the server when the target WebVR is terminated.

2) Data Processing and Training

The data collected from the VR-Sniffer is sent to the server for preprocessing. We found that unfolding the hexahedron and representing it on six separate 2D graphs gives each WebVR a unique shape. Figure 5a and 5b show examples of hit point graphs collected while running two target WebVRs, A-Blast and Moon Rider. A-Blast [25] is a shooting game using two VR controllers as guns, showing that most hit points are observed in the upward direction because target monsters appear mainly in the sky. On the other hand, the rhythm game Moon Rider [26] shows that most of the hit points are placed in the front and upward directions and spread left and right along the hand direction.

Based on this observation, we generate graph images that capture app-specific input features. For example, we apply a color gradation to hit points to indicate the clicking orders (shown in Figure 5c and 5d). In addition, VR users who play apps such as Jelly Face [27] and Trajectory Command [28] often press the button and drag it. As a result, hit points collected in the down- and up-click events that occur with a single click tend to be far away. To reflect this, we create a graph that connects hit point pairs for down- and up-clicks (shown in Figure 5e and 5f).

To determine the attack performance, we develop a deep learning model that classifies two types of graphs: a graph with only points for down-clicks (referred to as “down") and a graph with points for down- and up-clicks (referred to as “pairs"). The authors prepared a total of 96 training data by playing 12 target WebVRs 8 times per each. In order to overcome the limitations of time-consuming data collection, VR-Sniffer provides a method to augment data based on the collected data. The augmentation functions receive authors’ data as input and return augmented data by transforming hit points into the following rules:



(a) Hit points collected from A-Blast

(b) Hit points collected from Moon Rider

(c) Hit points with time information collected from A-Blast

(d) Hit points with time information collected from Moon Rider

(e) Hit points paired with down- and up-clicks collected from Jelly Face

(f) Hit points paired with down- and up-clicks collected from Trajectile Command

FIGURE 5: Hit point graphs generated by various feature extractions. The Empty planes are omitted.

$$< t_i, d_i, \mathbf{H}'_i > =< t_i, d_i, S \circ G(\mathbf{H}_i) > \quad (1)$$

$$G(\mathbf{H}_i) = \mathbf{H}_i + \alpha_g \mathcal{N}(0, \mathbf{D}_{d_i}), \quad (2)$$

$$S(\mathbf{H}_i) = \mathbf{H}_i + \alpha_s \mathbf{\Delta}, \quad (3)$$

$$\mathbf{\Delta} \sim \mathcal{N}(0, \mathbf{D}_{d_i}). \quad (4)$$

A hit point $\mathbf{H}_i$ is converted to $\mathbf{H}'_i$ by applying two transformation functions G and S. The first transformation G adds a noise vector to $\mathbf{H}_i$, where the noise is sampled from normal distribution $\mathcal{N}(0, \mathbf{D}_{d_i})$. $\mathbf{D}_{d_i}$ is a diagonal covariance matrix computed from $\mathbf{H}$ placed in the same plane from the data of a single user. The covariance matrix helps us make the transformation proportional to the coordinate variances of hit points. The second transformation S shifts $\mathbf{H}_i$ on a plane of d_i. The amount of the shift is $\alpha_g \mathbf{\Delta}$ where $\mathbf{\Delta}$ is sampled from the distribution $\mathcal{N}(0, \mathbf{D}_{d_i})$. We used $\alpha_g = 0.3$ and $\alpha_s = 0.15$ to control the amount of the transformations.

Note that G samples different noises, moving hit points to different ways, while S uses one noise to all hit points, shifting to one way. The coordinates of the hit point can vary even for the same UI button according to players with different heights and arm lengths, or clickable areas, these two functions make our model robust by reflecting such characteristics. On the other hand, other filters, such as symmetrical images, are not suitable for our model, so we deliberately excluded them. We empirically confirmed that ResNet [29] shows the highest performance by comparing the four well-known deep learning models for image recognition: LeNet [30], VGGNet [31], AlexNet [32], and InceptionNet [33]. In Section VI, we present the evaluation results for VR-Sniffer using ResNet in various aspects.

VI. EVALUATION

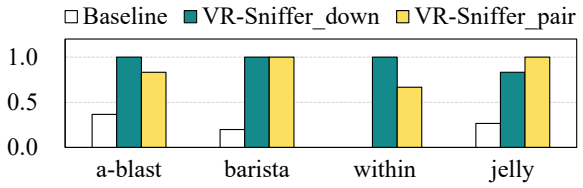
In this section, we describe the experimental setup and present the evaluation results for VR-Sniffer from two perspectives: (1) attack efficacy and (2) stealthiness.

A. EXPERIMENT SETUP

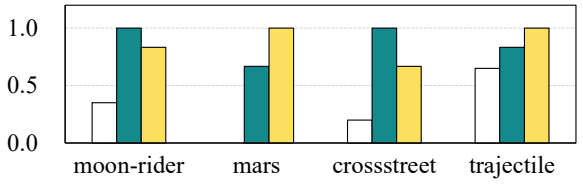
To investigate the user impact of the attack, we recruited 18 student participants from July 27 to August 2, 2022, with an average age of 25.4 years, including 6 females. All participants have VR experience, and three of them have experienced VR more than 10 times. We set up the space large enough to play WebVR and experimented using the HTC VIVE pro [34] and Firefox 102.0.1 on the Windows 10 host. Participants were asked to perform WebVR assignments and a survey according to our instructions. Since it is difficult to play all 12 VR apps, we divided the target apps into three groups (Group A, B, and C) and instructed each participant to play four each. As a result, a total of 72 test data were collected, 6 by each app. We explained how to play VR apps to the participants in advance and did not intervene during the VR play.

B. ATTACK EFFICACY

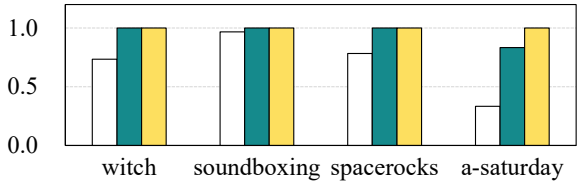
We first evaluate the attack performance with three metrics: classification accuracy for target WebVRs, TopK accuracy, and accuracy according to the size of the augmented data. Classification accuracy is calculated as the number of test data accurately classified among all test data, and TopK accuracy is calculated as the classification accuracy at the kth attempt on the test data that fails to identify. We compared the model performance trained with down and pair graph images with baseline performance trained with graph images presenting 3D coordinates of controller objects (see Figure 2) with the same network. VR-Sniffer is trained with augmented data described in Section V-B. There is no restriction on the attacker adding data because training deep learning models is a preliminary task that does not affect attack performance. We show the experimental result of measuring classification accuracy while increasing the size of the augmented data from 0 to 768 in Figure 8. In the remaining experiments, a total of 288 training data (96 data collected by running VR apps and 192 augmented data) were used.



(a) Classification accuracy of Group A participants



(b) Classification accuracy of Group B participants



(c) Classification accuracy of Group C participants

FIGURE 6: WebVR classification accuracy

Figure 6 shows the classification accuracy of target WebVRs for three approaches divided by user groups. Compared to baseline, VR-Sniffer performs well for all target WebVRs. The average accuracy was 40.4% in baseline, 93.1% in VR-Sniffer_down, and 91.7% in VR-Sniffer_pair, respectively. VR apps with high randomness of user input have lower classification accuracy than average, Access Mars in VR-Sniffer_down and Within and Cross the Street in VR-Sniffer_pair had the lowest classification accuracy of 66.7%, respectively. The performance of VR-Sniffer_down and VR-

Sniffer_pair shows complementary aspects. For example, apps, such as Access Mars that press the controller button with relatively constant intervals, show that the pair's information is effective, and apps, such as A-Blast that actively click, can perform better with a down technique that simplifies information. This result suggests that each of our techniques is sufficiently effective in itself and can exhibit greater effects when used contextually.

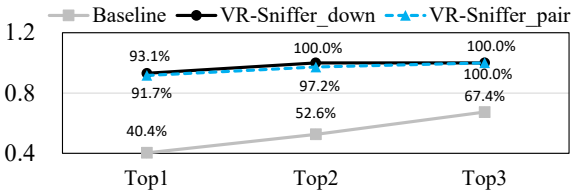


FIGURE 7: TopK classification accuracy

Figure 7 shows the average of the classification accuracy in the kth attempt for the three approaches. In all three approaches, the performance improves with repeated attempts. The baseline which has a large room for improvement shows great improvements as the attempts are repeated, however, the average classification accuracy of Top3 for 12 VR apps is 67.4%, which is difficult to say that it is able to identify the target VR apps. Our approach showed 100% classification accuracy in just a second attempt, except for A-Blast and Cross the Street, when using VR-Sniffer_pair. This result shows that VR-Sniffer can identify visiting webVRs with very high accuracy only in a few attempts.

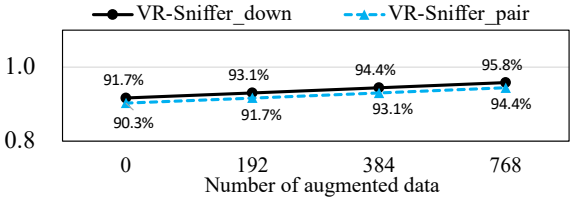


FIGURE 8: Classification accuracy by increasing augmented data size

Figure 8 shows the classification accuracy of VR-Sniffer according to the size of the augmented data. Each point is the result of augmenting the data by 0, 2, 4, and 8 for each training data, and the total number of training data is 96, 288, 480, and 864, respectively, by adding 96 of the original data. The result shows better performance as the augmented data size increases. As described in Section V-B, the coordinates of the hit point may vary for the same UI button depending on the user's body (e.g., height, arm length) or clickable area, this result indicates that our augmented technique captures such user-specific noise well. On the other hand, as the size of the training data increases, the training time increases. It took an approximately additional 19 minutes with 768 augmented data. However, model training is a task performed by an attacker in advance, which does not affect runtime inference and is independent of the performance of the attack. Therefore, this result shows that our approach can be further improved with the addition of learning data.

C. STEALTHINESS

In order to secretly complete an attack, VR-Sniffer must not affect the performance of target WebVR sites; otherwise, the victim can detect the presence of an attack. To verify this, we selected three VR apps, A-Blast, A Saturday Night, and Moon-Rider, and slightly modified them to measure 1) page loading time and 2) FPS in VR mode. The page loading time is a crucial performance indicator for websites that directly affects the number of visitors [35], [36]. Likewise, a sharp decrease in FPS reduces the user's immersion in VR mode, affecting WebVR performance [37].

TABLE 2: An average page loading time and FPS in attack and non-attack scenarios

	Page Loading Time (s)		FPS (drop rate)	
	normal	attacked	normal	attacked
A-Blast	379.1	388	89.97	89.56
A Saturday	266.6	321	80.22	80.07
Moon Rider	474.7	504.6	84.48	83.83

Table 2 shows the average page loading time and FPS performed 10 times in attack and non-attack environments for each of the three target VR apps. Compared to the non-attack environment, the average page loading time of three test VR apps increased by 31 msec and FPS decreased by 0.64, showing VR-Sniffer caused negligible performance overhead for the target WebVRs. This is because the attacker's WebVR running on the victim's computer is a simple VR app that renders a single cube object and does not need to be played in VR mode. We also remark that all participants answered in the interview that they were not aware of the attack at all. This shows that VR-Sniffer can infer target WebVRs in a stealthy manner.

VII. CONCLUSION AND DISCUSSION

In this paper, we devised a novel browsing history sniffing attack on WebVR that infers WebVRs currently visited by victims. We introduced a security flaw in the GamePad API that allows web attackers to capture sensor data on VR controllers and developed a technique to collect hit points and directions that express app-specific controller usage patterns. We also developed a WebVR classification model that is robust against random input sequences by converting the collected hit points into graph images. Our evaluation conducted with 18 participants showed that VR-Sniffer was able to successfully identify target WebVRs with 95.8% accuracy and had a negligible impact on their performance, enabling behave an attack stealthily.

Our study has a limitation in that it is not possible to identify VR apps without using a VR controller and without

any UI patterns. In addition, most of the WebVRs currently serving are in the game category, which tends to be less sensitive to personal information leakage. However, this work proactively warns of the new privacy threats WebVR faces by showing that the browsing history sniffing attack on WebVR is feasible with high accuracy. As a future study, we expect to be able to infer more sensitive information (e.g., watched video identification) performed by victims by analyzing user inputs from identified VR apps.

REFERENCES

[1] "WebVR 1.1," *World Wide Web Consortium (W3C)*. [Online]. Available: https://immersive-web.github.io/webvr/spec/1.1/. Accessed on: Sep. 06, 2022.
[2] E. W. Felten, and M. A. Schneider, "Timing Attacks on Web Privacy," *In Proceedings of the ACM Conference on Computer and Communications Security*, 2000.
[3] Z. Weinberg, E. Y. Chen, P. R. Jayaraman, and C. Jackson, "I Still Know What You Visited Last Summer: Leaking Browsing History via User Interaction and Side channel Attacks," *In Proceedings of the IEEE Symposium on Security and Privacy*, 2011.
[4] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The Web Never Forgets: Persistent Tracking Mechanisms in the Wild," *In Proceedings of the ACM Conference on Computer and Communications Security*, 2014.
[5] S. Chen, R. Wang, X. Wang, and K. Zhang, "Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow," *In Proceedings of the IEEE Symposium on Security and Privacy*, 2010.
[6] R. Schuster, V. Shmatikov, and E. Tromer, "Beauty and the Burst: Remote Identification of Encrypted Video Streams," *In Proceedings of the USENIX Security Symposium*, 2017.
[7] S. Lee, H. Kim, and J. Kim, "Identifying Cross-origin Resource Status using Application Cache," *In Proceedings of the Annual Network and Distributed System Security Symposium*, 2015.
[8] T. V. Goethem, M. Vanhoef, F. Piessens, and W. Joosen, "Request and Conquer: Exposing Cross-Origin Resource Size," *In Proceedings of the USENIX Security Symposium*, 2016.
[9] J. Lee, H. Kim, J. Park, I. Shin, and S. Son, "Pride and Prejudice in Progressive Web Apps: Abusing Native App-like Features in Web Applications," *In Proceedings of the ACM Conference on Computer and Communications Security*, 2018.
[10] D. Jackson, and J. Gilbert, "WebGL Specification," *Khronos Group*. [Online]. Available: https://www.khronos.org/registry/webgl/specs/latest/1.0/. Accessed on: Sep. 06, 2022.
[11] "GamePad API," *Mozilla Developer Network (MDN)*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Gamepad_API. Accessed on: Sep. 06, 2022.
[12] "WebXR Device API," *World Wide Web Consortium (W3C)*. [Online]. Available: https://www.w3.org/TR/webxr/. Accessed on: Sep. 06, 2022.
[13] "XRSession," *Mozilla Developer Network (MDN)*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/XRSession. Accessed on: Sep. 06, 2022.
[14] "Can I use WebXR Device API," *BrowserStack*. [Online]. Available: https://caniuse.com/?search=webxr. Accessed on: Sep. 06, 2022.
[15] "Can I use WebVR API," *BrowserStack*. [Online]. Available: https://caniuse.com/?search=webvr. Accessed on: Sep. 06, 2022.
[16] "iframe: Inline Frame Element," *Mozilla Developer Network (MDN)*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe. Accessed on: Sep. 06, 2022.
[17] "Raycaster," *three.js docs*. [Online]. Available: https://threejs.org/docs/#api/en/core/Raycaster. Accessed on: Sep. 06, 2022.
[18] "WebXR permissions and security," *Mozilla Developer Network (MDN)*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API/Permissions_and_security. Accessed on: Sep. 06, 2022.
[19] "WebVR Directory," *Supermedium*. [Online]. Available: https://webvr.directory/. Accessed on: Sep. 06, 2022.
[20] "The VR Comic Book Reader," *Supermedium*. [Online]. Available: https://supermedium.com/. Accessed on: Sep. 06, 2022.
[21] "About the Google Creative Lab," *Google Creative Lab*. [Online]. Available: https://thefwa.com/profiles/google-creative-lab. Accessed on: Sep. 06, 2022.
[22] "About Mozilla Labs," *Mozilla Lab*. [Online]. Available: https://labs.mozilla.org/projects/firefox-reality/. Accessed on: Sep. 06, 2022.
[23] "A web framework for building virtual reality experiences," *A-Frame*. [Online]. Available: https://aframe.io. Accessed on: Sep. 06, 2022.
[24] "A JavaScript 3D Library," *Three.js*. [Online]. Available: https://threejs.org/. Accessed on: Sep. 06, 2022.
[25] "A-Blast," *Mozilla VR*. [Online]. Available: https://aframe.io/a-blast/. Accessed on: Sep. 06, 2022.
[26] "Moon Rider," *Supermedium*. [Online]. Available: https://moonrider.xyz/. Accessed on: Sep. 06, 2022.
[27] Josh Shadik, "Jelly Face," [Online]. Available: https://jsh.sh/jelly-face/. Accessed on: Sep. 06, 2022.
[28] Adam Alexandr, "Trajectile Command," [Online]. Available: https://micosmo.com/trajectilecommand/. Accessed on: Sep. 06, 2022.
[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016.
[30] Y. LeCun, B. Léon, B. Yoshua, and H. Patrick, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE* 86, no. 11 (1998): 2278-2324.
[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *In The International Conference on Learning Representations (ICLR)*, 2015.
[32] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, 60(6), pp.84-90, 2017.
[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions," *In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1-9), 2015.
[34] "VIVE Pro Specs," *HTC*. [Online]. Available: https://www.vive.com/us/product/vive-pro/. Accessed on: Sep. 06, 2022.
[35] J. Manhas, "A Study of Factors Affecting Websites Page Loading Speed for Efficient Web Performance," *International Journal of Computer Sciences and Engineering,* vol. 1, 2013, pp. 32–35.
[36] F. F. NAH, "A study on tolerable waiting time: how long are Web users willing to wait?," *Behaviour and Information Technology,* vol. 23, 2004, pp. 153–163.
[37] J. E. Bos, W. Bles, and E. L. Groen, "A Theory on Visually Induced Motion Sickness," *Displays,* vol. 29, 2008, pp. 47–57.

JIYEON LEE received her B.Sc. degree in computer science from Dankook University, Republic of Korea, in 2012, and M.Sc. and Ph.D. degrees in school of computing from the Korea Advanced Institute of Science and Technology (KAIST), Republic of Korea, in 2014, and 2021, respectively. She is a post-doctoral researcher in the school of AI convergence, Soongsil University, Republic of Korea, where she joined in 2021. Her research interests include general topics in security/privacy and web vulnerability research.

BYUNGGILL JOE received B.Sc. and M.Sc. degree, respectively in 2015 and 2017, and currently pursuing his Ph.D. degrees in the same institute, school of computing from the Korea Advanced Institute of Science and Technology (KAIST), Republic of Korea. His research interests include general topics in adversarial robustness of machine learning methods and applying machine learning methods for healthcare applications.

KILHO LEE (Member, IEEE) is an assistant professor in the School of AI Convergence, Soongsil University, Republic of Korea. He received his BSc degree in Information and Computer Engineering from Ajou University, and his MSc and PhD degrees in Computer Science from KAIST. His interests include system design for real-time embedded systems and cyber-physical systems. He won the best paper award of ACM MobiCom 2019 and IEEE CPSNA in 2014.