I Can See You Brain: Investigating Home-Use Electroencephalography System Security

YINHAO XIAO, The George Washington University, USA
YIZHEN JIA, The George Washington University, USA
XIUZHEN CHENG, The George Washington University, USA
ZHENKAI LIANG, National University of Singapore, Singapore
ZHI TIAN, George Mason University, USA

Health-related IoT devices are getting more and more popular in recent years. On one hand, users can access information of their health conditions more conveniently than ever before with these devices; on the other hand, they are exposed to new security risks that are impossible to occur in traditional IoT devices. In this paper, we presented, to the best of our knowledge, the *first* in-depth security analysis on home-use electroencephalography (EEG) IoT devices. Our key contributions are twofold. First, we reversely engineered the home-use EEG system framework via which we identified the design and implementation flaws. By exploiting these flaws, we developed two sets of novel easy-to-exploit PoC attacks, which consist of four remote attacks and one proximate attack. In a remote attack, an attacker can steal a user's brain wave data through a carefully crafted program while in the proximate attack, the attacker can steal a victim's brain wave data over-the-air without accessing the victim's device on any sense when he is close to the victim. As a result, all the 156 brain-computer interface (BCI) apps in the NeuroSky App store are vulnerable to the proximate attack. We also discovered that all the 31 free apps in the NeuroSky App store are vulnerable to at least one remote attack via an empirical static binary analysis. Second, we proposed a novel deep learning model of a joint recurrent convolutional neural network (RCNN) to infer a user's activities based on the reduced-featured EEG data stolen from the home-use EEG IoT devices, and our evaluation over the realworld EEG data indicates that the inference accuracy of the proposed RCNN is as high as 70.55%, which significantly outperforms 11 other well-known learning models.

CCS Concepts: • **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

Additional Key Words and Phrases: Wireless sensor networks, media access control, multi-channel, radio interference, time synchronization

ACM Reference Format:
Yinhao Xiao, Yizhen Jia, Xiuzhen Cheng, Zhenkai Liang, and Zhi Tian. 2010. I Can See You Brain: Investigating Home-Use Electroencephalography System Security. *Proc. ACM Hum.-Comput. Interact.* 9, 4, Article 39 (March 2010), 27 pages. https://doi.org/0000001.0000001

Authors' addresses: Yinhao Xiao, The George Washington University, 2121 Eye Street, NW, Washington, DC, DC, 20052, USA, xyh3984@gwu.edu; Yizhen Jia, The George Washington University, 2121 Eye Street, NW, Washington, DC, DC, 20052, USA, chen2015@gwmail.gwu.edu; Xiuzhen Cheng, The George Washington University, 2121 Eye Street, NW, Washington, DC, DC, 20052, USA, cheng@gwu.edu; Zhenkai Liang, National University of Singapore, 21 Lower Kent Ridge Road, 119077, Singapore, liangzk@comp.nus.edu.sg; Zhi Tian, George Mason University, 4400 University Dr, Fairfax, VA, 22030, USA, ztian1@gmu.edu.

2573-0142/2010/3-ART39 $15.00
https://doi.org/0000001.0000001

1 INTRODUCTION

Health-related IoT devices have been constantly attracting public attentions in recent years. According to a most recent report, the market of healthcare IoT reaches 41.22 billion USD in 2017 and is projected to grow to 158.07 billion USD by the year of 2022 [12]. Besides the dramatic growing trend of the gross benefits, health-related IoT market is also prospering in the aspect of diversity of its products. Devices like smart wristbands and smart scales are two of the most well-known products in this area with which a user can monitor her health conditions varying from heartbeats, sleep level, to muscle mass or even bone mass.

On one hand, users gain tangible benefits from the health-related IoT devices. On the other hand, however, users are also exposed to new and unknown risks. In January 2017, Department of Homeland Security (DHS) confirmed that nearly 465,000 implanted heart pumping devices, which are actively used in hospitals all-round the US, are vulnerable to remote attacks [11]. For example, hackers can remotely hack into a patient's defibrillator and trigger irregular heart rhythms which can cause a cardiac failure [11]. In September 2017, DHS issued a warning stating that approximately 4,000 wireless syringe infusion pumping devices are vulnerable to remote attacks [21]. For instance, attackers can exploit a loophole to murder a patient by remotely giving the patient overdose infusion [21]. As one can see from these cases, different from compromising traditional IoT devices such as smart hues or thermostats that results in system failures, compromising health-related IoT devices can not only cause leakage of a user's health information, but also directly jeopardize the user's life.

Unlike the professional medical IoT devices mentioned above whose security problems have been gradually explored and addressed by more and more researchers, the security of home-use health-related IoT devices is seriously under-investigated. Therefore, a natural question is raised: are emerging home-use health-related IoT devices, which have a large amount of users, also suffer from similar security threats? To answer this question, we performed, to the best of our knowledge, the *first* security analysis on home-use electroencephalography (EEG) IoT devices. The reason we chose EEG devices for our research is because they have a drastic growing market which is projected to reach 1.40 billion USD by the year of 2025 [6]. Moreover, EEG data is one of the most important and sensitive human health data that can reveal an individual's sensitive health conditions. Martinovic *et al.* [40] showed that it is completely possible for an attacker to guess a user's password through EEG data. Therefore, it is urgent and critical to investigate the security vulnerabilities of home-use EEG devices.

In this paper, we studied the security of home-use EEG devices targetting the ThinkGear AM (TGAM) module manufactured by NeuroSky. NeuroSky is the most well-known manufacturer of the home-use EEG devices since it is the first company to make brain wave devices for home use in the world [15], and it still holds the largest market share of home-use EEG devices [2]. TGAM is the exclusive brain wave sensor ASIC module developed by NeuroSky; it was elected as TIME Magazine's 100 Best Toys of All Time and has a potential to be widely adopted by home-use EEG IoT manufacturers [8]. In this research, we demonstrated that based on the security flaws of the EEG system framework due to the coarse-grained implementation, an attacker is able to construct two sets of easy-to-exploit PoC attacks, which contain 4 remote attacks and one proximate attack, to actively steal a user's brain wave data. A remote attack does not require the attacker to be close to the victim but a carefully crafted malicious program is needed. The proximate attack, on the other hand, does not require any malicious program, but it has to be launched within a certain distance away from the victim. Because the security flaw exploited by our proximate attack exists in a mandatory step for the EEG data retrieval, all the 156 brain-computer interface (BCI) apps appearing in the NeuroSky App store, the official App store for TGAM devices, are vulnerable to this attack. We also conducted an empirical static binary analysis against all the 31 free apps in the NeuroSky App store and found that all of them are vulnerable to at least one remote attacks.

In addition, we studied the potential privacy leakage problem from the EEG data collected by the TGAM devices. It is well researched that rich-featured EEG signals collected by strict medical-use devices or research-use devices can reveal a user's sensitive information, e.g., focal disorders [7] [29] and sleep levels [42]. However,

whether or not reduced-featured EEG data collected by home-use EEG devices can also pose similar privacy threats remains unexplored. In this paper, we proposed a novel recurrent convolutional neural network model (RCNN) targeting the reduced-featured EEG data collected by home-use EEG IoT devices to infer a user's focusing activities; our evaluation results over the realworld EEG data revealed that the proposed RCNN has an accuracy as high as 70.55%, which significantly outperforms the other 11 most widely-used learning models.

Finally, we proposed three easy-to-adopt defense solutions to eliminate our proposed attacks. Two defense mechanisms were devised for the remote attacks and the other one was devised for the prosimate attack.

Our Contributions. In summary, we have the following contributions in this paper:

- To the best of our knowledge, we proposed the first security analysis against home-use EEG IoT devices. We demystified the NeuroSky EEG system framework, which is not documented by the official NeuroSky documentations.
- We identified the security flaws in the NeuroSky EEG system framework, based on which we constructed and fully implemented two sets of easy-to-exploit attacks to steal a user's brain wave data. We also analyzed the BCI apps on the NeuroSky App store and discovered that all apps are vulnerable to the proximate attack, and that all the free apps are vulnerable to at least one remote attack.
- We also proposed a novel deep learning model to infer a user's activities based on the stolen reduced-featured EEG data with an inference accuracy as high as 70.55%, which significantly outperforms other well-known learning models such as SVM, decision tree, and adaboost.
- We finally proposed defense solutions to eliminate our propsed attacks. The solutions are effective and are easy to be adopted.

Paper Organization. The rest of the paper is organized as follows. Section 2 outlines the most related work. Section 3 introduces the preliminary knowledge on EEG, the types of different EEG devices, and our threat model. Section 4 details the demystified EEG system framework. Section 5 demonstrates the security flaws of the framework and the implementations of our attacks based on these flaws. Section 6 presents our deep learning model for the reduced-featured EEG inference attack. Section 7 reports the performance of our attacks and Section 9 concludes the paper.

2 RELATED WORKS

With the drastic development of home-use IoT systems, more and more studies were carried out to investigate the vulnerabilities of and attacks on smart home systems; on the other hand, research on sensitive personal information leak by the smart home IoT systems, especially the smart health systems, just started to thrive. In this section, we provide a brief overview on the most related research.

Health-related IoT Security. The security of health-related devices and systems have gained more and more attention in recent years due to the rapid development and the tight personal health connection. Eberz *et al.* [30] presented a systematic attack against the ECG biometrics and demonstrated its effectiveness by applying it to a successful commercialized ECG biometric product, the Nymi Band; they transformed the ECG signals collected on different ECG devices by a mapping function such that the signal collected on one device can be mimicked by the signal collected by a different device so that they can use arbitrary ECG signals for spoofing attacks. Different from their work, we focused more on the vulnerabilities of the home-use NeuroSky EEG system framework and the leak of the sensitive information revealed by the raw EEG signals; we developed 5 PoC attacks through which victim's activities can be surmised by analyzing the captured raw EEG signals.

Rahman *et al.* [43] investigated the security and privacy issues of Fitbit; after reversely engineering the ANT protocol for data communications, they generated various attacks including the data capture attack, injection attack, denial of service attack, and so on; they also proposed possible defense mechanissms against the proposed attacks. Li *et al.* [38] demonstrated several security attacks on a popular glucose monitoring and insulin delivery

system available on the market; by recovering the radio protocol, they proposed various attack scenarios and performed two types of attacks, passive attack and active attack; then they analyzed the attack scenarios and generated two defense schemes. Compared to the two pieces of research work mentioned above, besides the security of the health-related device itself, we also focused on the sensitive information leakage revealed by the captured EEG signals.

Martinovic *et al.* [40] explored a research-use EEG devices, EPOC, as a potential attack intermediary to infer private information about their users; with the EEG signals captured from EPOC, they calibrated the classifiers on a set of training observations; their results supported the hypothesis that personal sensitive information such as bank account and password might be unintentionally leaked. In contrast to their research, we chose a much more popular setting with a home-use EEG IoT device, which has much fewer electrodes than the ones of EPOC; instead of using the EEG device as an attack intermediary, we directly explored the security vulnerabilities of the EEG device itself; based on the weaknesses, we implemented 5 PoC attacks to steal the EEG data from the device; with the reduced-featured EEG signals, we can infer the user's activity at approximately 70.55% accuracy.

Alongside the security study on health-related IoT devices, many work focused on building smart health systems and platforms. In [46] [49], the authors discussed the concept of health-related smart city and the home-based wellness platform. Mirza *et al.* [25] provided an overview on the design and modeling of the current smart health monitoring systems. Coincidentally, ISLAM *et al.* [35] investigated the e-Health technologies and reviewed the existing advanced network architectures for e-Health; they also analyzed the distinct security and privacy features of the e-health structures, proposed an collaborative security model, discussed the innovations in health care contexts, and addressed various e-health policies and regulations.

More General IoT Security. General IoT security research has been thriving in recent years. Fernandes *et al.* [31] studied the Samsung's SmartThings smart home system by performing a combination of static analysis, runtime testing, and manual analysis on a dataset of 499 SmartApps and 132 device handlers downloaded in source form; they discovered 2 design flaws that lead to over-privileges even with the privilege separation module implemented, and found that the SmartThings event subsystem does not sufficiently protect the events that carry sensitive information; by exploiting these design flaws, they developed 5 PoC attacks targetting the SmartThings smart home system. Costin *et al.* [27] managed to unpack and analyze 26,275 embedded system firmware images crawled from the Internet; with static analysis, they discovered a total of 38 previously unknown vulnerabilities in over 693 firmware images. Müller *et al.* [41] conducted a large scale analysis on the known printer attacks, and provided a general method for security analysis on printers; they also wrote a prototype software PRET to automatically evaluate 20 printer models and found that all of them are vulnerable to at least one of the known attacks. Manos *et al.* [24] did a comprehensive analysis on Mirai's emergence and evolution. They analyzed how the botnet emerged, what classes of devices were affected, how Mirai variants evolved and competed for vulnerable hosts, and what types of attacks Mirai exploited. Flavio *et al.* [32] found that the entry systems of most VW Group vehicles maintain a few global master keys, which allows an adversary to clone a remote control and gain unauthorized access to a vehicle; they also presented a novel correlation-based attack on the Hitag2 rolling code scheme, which reveals the cryptographic key and thus enables an attacker to clone the remote control system. In contrast, our work focuses on the vulnerabilities of a specific type of home-use EEG system and its related sensitive information leakage.

There also exsits research targetting the communication protocols of the IoT systems. For example, Ronan *et al.* [44] discovered a bug in the Touchlink part of the ZigBee Light Link protocol implemented by Philips, and compromised the global AES-CCM key used to encrypt and authenticate a new firmware so that an attacker can replace the benign firmware with a malicious one over-the-air; with all these findings they implemented a smart light bulb worm which automatically spreads and controls all the Philips smart lights. Our work is along with a similar line except that we further analyzed the captured EEG signals rather than just taking full control of the devices.

3 BACKGROUND

In this section, we first introduce the preliminary knowledge about electroencephalography (EEG). Then we briefly cover the differences among medical-use EEG devices, research-use EEG devices, and home-use EEG devices. Lastly, we demonstrate our threat model for the home-use EEG devices we are going to investigate.

3.1 Background Knowledge of Electroencephalography

Human brain is consisted of thousands of nerve cells, also known as neurons. These neurons are constantly receiving, processing, and propagating information through electrical or chemical signals. The neurons are densely interconnected via a nervous structure with numerous synapses serving as intermediaries for passing or inhibiting electrical or chemical signals from one neuron to another neuron. There are two types of synapses: chemical synapse and electrical synapse. A chemical synapse converts an electrical activity from the pre-synaptic neuron into a chemical substance called neurotransmitter which is passed to the post-synaptic cell. Different from the chemical synaptic system, in the electrical synaptic system, the pre-synaptic neuron passes an electrical activity to the post-synaptic neuron through a special channel called gap junction where the electrical activity is converted into an ionic current and therefore inducing voltage changes between the pre-synaptic cell and the post-synaptic cell. Note that electrical synapses transmit signals much more rapidly than chemical synapses, and some synapses can act as both an electrical and a chemical synapse. Electroencephalography (EEG) is an electrophysiological method to detect and monitor the electrical activities of the generated voltages in the electrical synaptic system. Even though the electrical current generated by a single synaptic activity may be too tiny to be detected, fortunately, any human cerebral activity such as surprise, alertness, anger, and so on, usually requires more than 1,000 synaptic activities to complete, making EEG recording, which relies on physical measuring with electrodes attached to a scalp, possible.

Even though EEG was introduced almost 80 years ago [48], its potential is still actively researched nowadays. In clinical contexts, it has already been well known that EEG can help diagnose various focal disorders such as epilepsy, head injury, encephalitis, brain tumor, encephalopathy, memory problems, sleep disorders, stroke, and dementia [7]. In recent years, the ability of EEG to predict Alzheimer's disease is triggering more and more attention [29]. In application contexts, besides the various games [28], EEG has been actively studied for the fields such as sleep monitoring [42] or even texting [50]. On one hand, human beings can benefit from advancing technologies on EEG analysis; on the other hand, as one can readily see, if an individual's EEG data is not properly secured, it can cause severe threats to the individual's privacy.

3.2 Medical-Use, Research-Use and Home-Use EEG

According to our research, the current EEG methods and their corresponding devices can be classified into three categories based on the uses of EEG: for strict medical use, for research use, and for home use.

3.2.1 Strict Medical-Use EEG. The first type of EEG is intended for strict medical-use, which is mainly employed for focal diagnosis. The EEG data collected for this purpose usually has high dimensions of features. For example, traditional devices such as the large array magnetoencephalography system (MEG) employed by typical hostipals, has up to 125 electrodes attached to a patient's scalp for EEG data collection, which implies that each sample of the EEG data has at least 125 dimensions of features [37]. Since these types of devices are intended for the extreme professional use only, they are usually not available in the markets for purchase.

3.2.2 Research-Use EEG. This type of EEG is intended mainly for research use, e.g., BCI research, which is often considered as a result of donwsampling the strict medical-use EEG. One of the most well-known research-use EEG systems is the BCI2000 [45]. BCI2000 is a general purpose software system for brain-computer interface research whose device uses 64 electrodes covering a scalp to monitor EEG, which means that each collected data

sample has at least 64 features. Even though BCI2000 devices are not intended for strict medical use, they are still classified as professional devices which are costly and are not available for purchase publicly. A similar system is EPOC [47] whose device has 14 electrodes attached to a scalp. This device again is not available for purchase in the markets.

3.2.3 Home-Use EEG. Home-use EEG is intended mainly for home-use applications, i.e., for lightweight medical-related uses. The device of this type of EEG only has three electrodes, with one attached to a person's left earlobe, one attached to the person's left forehead, and the last one attached to the person's right temple. The market of this type of EEG is dominated by a company called NeuroSky [18], which is the first company to commercialize home-use EEG devices and maintains the largest home-use EEG market share so far [2, 15]. NeuroSky invented the PCB module, i.e., the ThinkGear AM (TGAM), which serves as a multi-mode sensor to collect EEG data, and the EEG system framework for EEG data transmission. NeuroSky has an App store where users can purchase or download BCI apps. It also provides a software development kit (SDK) for developers to build BCI apps. All in all, NeuroSky establishes a mature platform for commercialization of the home-use EEG, and hence becomes the most popular brands in home-use EEG. Even though there exist other manufacturers for home-use EEG devices, most of their products leverage TGAM and adopt the EEG system framework. Compared with the previous two types of EEG devices, home-use EEG devices are much more affordable and are publicly available in many online markets such as Amazon.

TGAM measures and offers 10 features of EEG, namely attention meter, mediation meter, delta, theta, low alpha, high alpha, low beta, high beta, low gamma, and high gamma. Besides attention meter and mediation meter which are calculated by previously studied algorithms, the other 8 types of data are traditional EEG waveforms and can be directly measured through different frequencies.

- **Delta:** Delta is measured with frequency less than 4 Hz. It is often found during sleeping.
- **Theta:** Theta is measured with frequency ranging from 4 Hz to 7 Hz. It is often found during relaxation and meditation.
- **Alpha:** There are two types of alpha: low alpha which is measured with frequency ranging from 8 Hz to 9 Hz and high alpha which is measured with frequency ranging from 10 Hz to 12 Hz. Alpha is usually detected when eyes are closed or during relaxation.
- **Beta:** There are two types of beta: low beta which is measured with frequency ranging from 13 Hz to 17 Hz and high beta which is measured with frequency ranging from 18 Hz to 30 Hz. Beta is usually found during alertness or focuses.
- **Gamma:** There are two types of gamma: low gamma which is measured with frequency ranging from 31 Hz to 40 Hz and high gamma which is measured with frequency ranging from 41 Hz to 50 Hz. Gamma is usually associated with multi-sensory processing, e.g., perception involving different senses such as sound and sight.

Note that NeuroSky produces two types of EEG headsets: one is intended for personal computer (PC) only and the other can be used for both PC and mobile devices such as Android and iOS. However, we notice that the headset intended for PC only is much more popular than the one intended for both PC and mobiles since the purchase number of the former far exceeds the latter both on Amazon and Google Shop, and all the 156 apps on the NeuroSky official App store have a PC version but only 45 (25.85%) of them have the mobile version. On the other hand, the mechanism adopted by the mobiles is basically the same as the one adopted by PC. Therefore, in this paper, we conducted all our analysis against the PC platform since it is more meaningful.

3.3 Threat Model

In our adversarial model, attackers aim to steal brain waves of NeuroSky users and infer sensitive personal activity from these brain waves. For our remote attacks, we assume that an adversary is able to install a malicious

program into a victim's PC and to execute the malicious program. There exist various ways for the adversary to accomplish this task. For example, the adversary can perform social engineering [22], or exploit the known remote code execution (RCE) vulnerabilities [19] and advanced persistent threat (APT) [5]. Note that these techniques are all viable and feasible, and have been widely adopted as attack vectors for compromising industrial systems [4][3][1]. In this paper, we do not detail these procedures since they are out of the scope of our research. On the other hand, our proximate attack requires an adversary to be in a short-distance range, i.e., no more than 22 meters, away from the victim. However, the adversary does not need to have any access to any of the victim's devices. An adversary exploiting this attack can be a malicious neighbor, or a stalker who approaches the victim's home. If the victim wears the EEG headset device in a public area along with his laptop, the adversary can simply launch the attack in a nearby area, which could be some distance away and blocked by a few layers of walls, without triggering his suspicion. The adversary only needs to purchase related devices such as a GNU radio for a few hundred dollars.

4 DEMYSTIFYING THE EEG SYSTEM FRAMEWORK

Before proceeding to introduce our new attack vectors, we first investigated the system framework adopted by the NeuroSky devices. As we mentioned in Section 3, most NeuroSky devices leverage TGAM, which employs the EEG system framework. Unfortunately NeuroSky does not explicitly publicize the specification details of the framework; therefore we conducted an empirical study by both reading the related documents such as the development guide and user guide published publicly, and conducting dynamic testing cases, to thoroughly recover the EEG system framework. We found that the EEG system framework can be separated into two sides, the user application side where high-level BCI apps reside and the headset side where brain waves are collected and transmitted.

4.1 User Application Side

In the EEG system framework, the software-level implementations and operations are all integrated in the user's application side, whose hardware devices include the user's PC where BCI apps are running on and a radio frequency (RF) dongle serving as a communication bridge between BCI apps and the EEG headset, which is attached to the PC via a USB port. After exploring the developer's documentation, we recovered the sub-framework in the user application side and found that a BCI app running in the user's PC has the following two ways of retrieving the brain wave data: either from the standard software development kit (SDK) which retrieves the requested data from the RF dongle through the USB serial port, or via a pre-defined low-level socket protocol called ThinkGear socket protocol (TGSP).

4.1.1 EEG Data Retrieval through Standard SDK. The EEG system framework provides a standard SDK for Windows operating systems so that Windows-based BCI apps can directly invoke API calls provided by the SDK for EEG data retrieval and all the necessary API functions are offered by a dynamic link library (DLL) binary file called `thinkgear.dll`. Unfortunately, the documentation lacks a systematic demonstration of the APIs in the DLL; therefore we reversely engineered the DLL and found that besides the DLL main entry function `DLLEntryPoint`, it exports 19 other API functions. After further exploration, we identified 4 functions that are mandatory to complete a EEG data retrieval; the other 15 functions are optional as they perform tasks such as writing a log stream, setting a baud rate, and reading a driver's version. The 4 mandatory functions are `TG_GetNewConnectionId` through which the API caller is assigned an available integer as an identifier for a new connection to the RF dongle serial port, `TG_Connect` which initializes a new connection that is bound with the provided connection id, `TG_ReadPackets` which reads a raw packet through the RF dongle serial port, and `TG_GetValue` which parses and returns a requested EEG data from a raw packet. The calls have to be made in the order of `TG_GetNewConnectionId`, `TG_Connect`, `TG_ReadPackets`, and finally `TG_GetValue`.

4.1.2 EEG Data Retrieval through TGSP. The EEG system framework also provides a low-level socket protocol, TGSP, for EEG data retrieval. TGSP can not only simplify the EEG data retrieval process since BCI apps do not need to go through the tedious SDK API calls, but also allow BCI apps to be implemented on different platforms other than Windows. TGSP adopts a server-client structure where there is an official server provided by NeuroSky, called ThinkGear Connector (TGC), which runs in the user's Windows platform, and BCI apps serve as clients sending requests to TGC and retrieving the desired data.

The communications between TGC and the RF dongle is not complicated. As a matter of fact, the method leveraged by TGC to retrieve the EEG data from the RF dongle is fundamentally identical to the standard SDK; however, TGC integrates the API functions inside the TGC program with Windows .NET framework instead of making API calls to `thinkgear.dll`. The communications between TGC and BCI apps follows the following procedure: TGC first opens a fixed TCP port number `13854`, then a BCI app verifies itself with the TGC by sending a JSON request with the format shown below:

```
{"appName": "some app name", "appKey":"some app key"}
```

where the attribute `appName` is the name of the BCI app and `appKey` is a SHA-1 key of `appName` according to the documentation. Normally, the `appKey` should serve as a verification code for authentication. However, after a further exploration we noticed that the `appKey` serves merely as the app's identifier instead of a secret key assigned by the server; therefore, the whole verification process has little security implication.

After the verification process, the BCI app can send a request with the following format to retrieve the EEG data from TGC:

```
{"enableRawOutput": true/false, "format": "Json/BinaryPacket"}
```

where `enableRawOutput` indicates whether or not TGC should return the raw voltage data from the three electrodes and `format` specifies whether TGC should return the data in the format of JSON or as a binary raw packet.

If requested with the JSON format, TGC responds with the format shown below:

```
{"poorSignalLevel":n1,"eSense":{"attention":n2,"meditation":n3},"eegPower":
   {"delta":n4,"theta":n5,"lowAlpha":n6,"highAlpha":n7,"lowBeta":n8,"
   highBeta":n9,"lowGamma":n10,"highGamma":n11}}
```

where `poorSignalLevel` specifies the poorness level (ranging from 0 to 200, with 0 indicating a good signal while 200 indicating the off-head state) of the signal resulted from how properly the user wears the headset.

If requested with the `BinaryPacket` format, TGC responds with raw packets following the structure shown in Table 1. A packet begins with two `SYNCS` of value `0xAA` for each, followed by one byte indicating the length of the payload, the payload itself, and the 1-byte checksum for the payload. The payload is stacked with a number of `DataRows`. A `DataRow` is a special structure shown in Table 2 with a size of PLENGTH ranging from 1 to 256 bytes. It begins with an optional segment called `EXCODE` valued `0x55` with undefined size (undefined number of `0x55`s), followed by the 1-byte `CODE` indicating what type of data is passed, an optional 1-byte segment `VLENGTH` indicating the length of the value, and finally the value itself. The choices of `CODE` values are documented in [23]. Some of the most important ones include `0x02`, `0x04`, `0x05`, and `0x81`, representing poorness level of the signal, attention meter, meditation meter, and EEG power, respectively.

4.2 Headset Side

In the EEG system framework, the TGAM chip in a headset collects user's EEG signals through the 3 electrodes and transmits the signals over-the-air through software defined radio (SDR). SDR is a novel radio technology in

Table 1. TGSP Raw Packet Structure

	SYNC	SYNC	PLENGTH	PAYLOAD	CHECKSUM
Content	`0xAA`	`0xAA`	Length of payload	Series of `DataRows`	Checks if the payload has bit errors.
Size	1 byte	1 byte	1 byte	PLENGTH	1 byte

Table 2. Raw `DataRow` Structure

	(EXCODE)	CODE	(VLENGTH)	VALUE
Content	`0x55`	Type of data	Length of Value	Value of the data
Size	Undefined	1 byte	1 byte	VLENGTH

which the traditional components within the radio communication system, e.g., mixers, filter, modulator, and demodulator, which are used to be implemented in hardware, are realized by software. In our case, the TGAM in the headset encodes the EEG raw packets into radio signals, applies one types of modulation to the signals, and finally propagates the signals at a certain range of frequency; while the RF dongle takes care of receiving the signal from the predetermined frequency, filtering noises, demodulating signals, and finally decoding the EEG raw packets. Even though the SDR specification is not included in the user manual guide provided by NeuroSky, fortunately, we found that the Federal Communication Commission (FCC) ID of the NeuroSky home-use EEG IoT device, "XG9MW1", is printed on the device headset. According to per U.S. law, all products which emit radio waves have to register with FCC to maintain test reports of radio specifications. After exploring the headset's test report [17] by searching the FCC ID, we found that the TGAM chip can operate on the frequency band ranging from 2419.9 MHz to 2470.9 MHz, and the modulation type is minimum shift keying (MSK) where messages are encoded into a sinusoid wave and a bit 0 and a bit 1 differ by half of a carrier period.

However, as pointed out by the test report, a ThinkGear EEG device has an operable frequency range of 51 MHz, which is too large for feasible analysis. In order to find the fine-grained center frequency via which the device is operating, we employed HackRF [10], a SDR peripheral device capable of transimiting and receiving radio signals from 1 MHz to 6 GHz, and GQRX [9], an open source SDR receiver powered by the GNU Radio. We conducted 30 experiments on 3 different NeuroSky EEG devices, 10 for each, and found that all devices in all experiments used the same center frequency with the same bandwidth: the approximate center frequency is 2458.4 MHz with an occupied bandwidth of approximately 1 MHz. A sample of the TGAM SDR wave is shown in figure 1.

According to our experimental study and analysis described above, we concluded that the entire EEG system framework can be sketched as in figure 2. It is consisted of four components: BCI apps, RF dongle, TGSP server,

and TGAM. A BCI app is installed in the user's application side, and it has two ways of retrieving the user's EEG data when running, via either standard SDK or TGSP. If the BCI app requests an EEG data via the standard SDK, it directly communicates with the RF dongle through the USB serial port which returns the raw binary packet of the EEG data; if the BCI app requests an EEG data via the TGSP server (officially, the Thinkgear Connector), the server then retrieves the EEG data from the RF dongle likewise. Lastly, through SDR, the RF dongle receives the raw EEG data collected by the headset with the multi-modal sensor TGAM which employs three electrodes to measure the brain waves from a human scalp. The SDR has a center frequency roughly located at 2458.4 MHz with an occupied bandwidth of approximately 1 MHz.

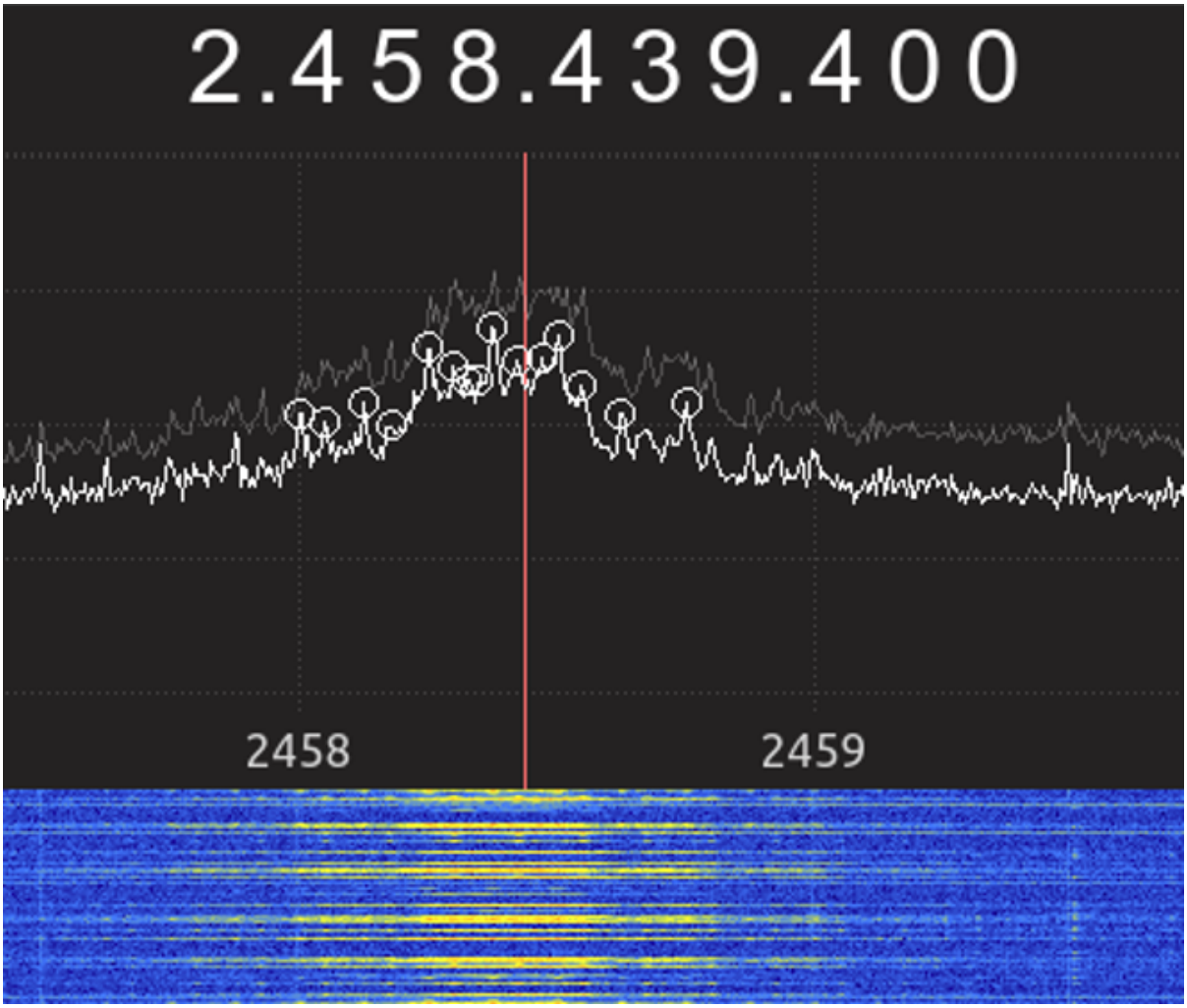


Fig. 1. A sample of the TGAM SDR wave. The center frequency is roughly 2458.4 MHz and the occupied bandwidth is approximately 1 MHz.

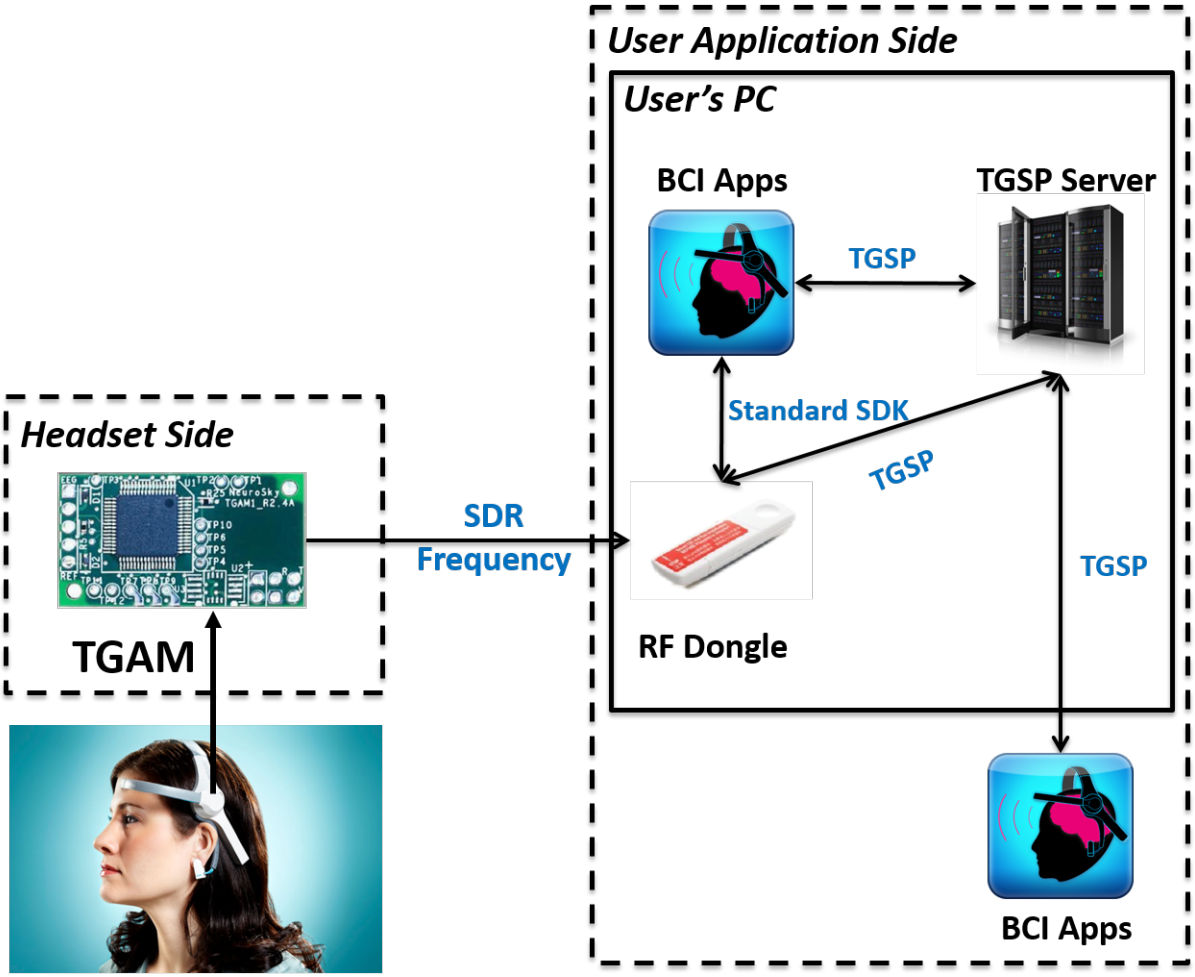


Fig. 2. The graph of the entire EEG system framework. A BCI app runs on a user's PC or other platforms within the user application side. If running on the user's PC, the BCI app can retrieve user's EEG data via either the standard SDK or TGSP. If running on a platform other than the user's PC, a BCI app can only retrieve the EEG data via TGSP. Both SDK and the TGSP server communicate with the RF dongle, which retrieves the EEG data from the user's headset through a SDR network.

5 ANALYSIS AND IMPLEMENTATION OF ATTACK VECTORS

In this section, we detail our novel attack vectors targeting the EEG system framework we demystified in Section 4. Our attack vectors can be classified as either remote attacks or proximate attacks. In a remote attack, an attacker can launch the attack without any constraint on how far he is away from the victim; however, a carefully crafted malicious program residing in the victim's PC side *is* required to run. In a proximate attack, on the other hand, an attacker has to launch the attack within a certain distance away from the victim; however, *no* malicious app is needed in this case.

5.1 Remote Attacks

We came up with four remote attack cases based on the roles a malicious program plays in the EEG system framework: as a malicious BCI app, as a malicious TGSP server, or as a malicious SDK. We have fully implemented all these attacks.

5.1.1 Malicious Program as a BCI App. The most straightforward attack is to install a malicious BCI app which secretly steals a victim's EEG data. As described in Section 4, a BCI app can retrieve EEG data through either the standard SDK or TGSP. Correspondingly, there could be two approaches to implement such a malicious program.

Exploiting Standard SDK: We first analyzed if it is possible to implement a malicious BCI app which steals the victim's EEG data by exploiting the standard SDK. After an initial exploration, we found that if there is no other BCI apps running, our malicious app can successfully retrieve the EEG data; however, if there is another BCI app running, this method fails. We then conducted dynamic debugging on the binary `thinkgear.dll` and found that when a BCI app calls `TG_Connect`, `TG_Connect` calls a sub function which invokes `CreateFileA` in Windows `kernel32.dll` and passes the fixed name of the RF dongle serial port, e.g., "COM3", as the file name to `CreateFileA`. Therefore, when a BCI app is accessing the serial port through SDK, another app would fail since `CreateFileA` would return the `ACCESS_DENIED` error. Hence, through this method, a malicious app would have to terminate the benign program running a BCI app or the TGSP server (which is basically a BCI app), if any, in order to accomplish the attack. Intuitively, terminating a running benign app can trigger a victim's suspicion. Hence, a malicious app has to imitate the terminated benign app in every level, imposing a tremendous burden to the attacker.

Exploiting TGSP: A relatively easy approach is to exploit TGSP; thus we examined the feasibility of a malicious BCI app accessing the EEG data through TGSP. As we mentioned in Section 4, the TGSP server adopts a naive verification step merely for the purpose of distinguishing different apps. Therefore, we implemented a malicious app using C/C++ which generates a random string as the app name, constructs a SHA-1 digest for the app name, and sends these two pieces of information to TGC. If there is a conflict, our malicious app simply re-generates another random app name. It turns out that this attack functions normally.

5.1.2 Malicious Program as a TGSP Server. Instead of creating a malicious BCI app, we also created a malicious program which acts as a fake TGSP server and responds to benign BCI apps while retrieving the EEG data using the standard SDK. Note that even though this attack is refined based on the previous attack, it is more complicated since it needs to maintain TGSP so that benign BCI apps can run normally.

In order for this attack to operate successfully, we first need to terminate the legitimate TGSP server, TGC, if it is running. TGC is not executed at the administrator level so that it can be terminated easily by calling the Windows Kernel API `TerminateProcess`. Then our malicious server maintains the TGSP specifications as we described in Section 4. This attack is fully implemented in C/C++.

5.1.3 Malicious Program as SDK. Besides the attacks mentioned above, we also constructed an attack by modifying the standard SDK into a malicious one. In Section 4, we identified the 4 mandatory API functions in `thinkgear.dll` to complete the EEG data retrieval. Among these 4 functions, the most important one is `TG_GetValue` since it returns the requested EEG data. Hence, directly attacking this function in `thinkgear.dll` is the simplest and most effective way to steal the EEG data.

In order to modify `TG_GetValue`, we had to dig into the detailed specification of the function. `TG_GetValue` is defined in the `thinkgear.h` header file; it has two parameters: an integer `connectionId` and an integer `dataType`, and returns a float value of the EEG data specified by `dataType`. We were interested in `dataType` and the returned float value while the `connectionId` has little implication to us. The `dataType` has 13 different values, namely the 8 EEG waveforms as we introduced in Section 3, the raw voltage value, the meditation meter, the attention meter, the signal poorness level, and the remaining battery power.

Having identified our interested values in the function declaration, we next looked into the function body written in the `thinkgear.dll` to come up with a way to retrieve these values. As one can see in the binary codes of the function `TG_GetValue` for a successful data return shown in figure 3, the value of `dataType` is passed to `EAX` and the returned value is pushed onto the FPU floating point register stack `st0`; therefore, in order to steal these two values, we need to conduct a DLL injection attack.

In Windows operating systems, there are basically two types of DLL injection attacks: dynamic DLL injection and static DLL injection. In a dynamic DLL injection attack, the malicious DLL is injected at runtime when the target process is running. In a static DLL injection attack, an attacker can simply replace the benign DLL with a carefully-crafted DLL file. Under certain circumstances only dynamic injection can be applied, e.g., the target DLL has system-level privilege which prevents a third-party program from directly modifying it; and in other cases, static injection may be possible. Static injection is more effective since dynamic injection requires the target process to be running, and the malicious program has to know the process id of the target process. In our case, `thinkgear.dll` does not require the system-level privilege; thus we used static injection for our attack.

Having decided to use static DLL injection, we then dug into the details of altering the benign `thinkgear.dll` into a malicious one. As discussed above, we need to steal two values, `EAX` and `st0`; therefore, we crafted a shellcode that can create a TCP client socket via which these two values are passed to our malicious server. In addition, after looking into the PE header of `thinkgear.dll`, we found that the virtual size and the raw size of the `.text` section are 70,692 (`0x11424`) bytes and 71,168 (`0x11600`) bytes, respectively, meaning that we need to squeeze the size of our shellcode to be less than 476 bytes since otherwise we have to either put the shellcode into other sections such as `.data`, which may fail to be executed due to data execution protection (DEP), or modify the PE header of `thinkgear.dll`. Fortunately, we were able to implement a shellcode with a size of 275 bytes, which can steal the two values and pass to our malicious server via a TCP connection. The address for injecting the shellcode begins at `0x11824` since the raw offset for `.text` is `0x400` (`0x11824=0x11424+0x400`). The injected shellcode is shown in figure 4. Lastly, we put an unconditional jump instruction to jump into the address of our shellcode. We modified the instruction `retn` to `jmp` then `retn`. Note that since static injection directly modifies the raw PE file other than the virtual memory, address space layout randomization (ASLR) does not have any effect on the `jmp` instruction.

Note that one can also attack `TG_ReadPackets` following a similar procedure as that for attacking `TG_GetValue`. However, `TG_ReadPackets` returns the raw packet values with redundant information such as `SYNC` and `CODE`. Therefore, directly attacking `TG_GetValue` is more efficient.

5.2 Proximate Attack

Different from the remote attacks in which a malicious program is required, launching a proximate attack does not require installation of any malicious program; however, a proximate attack requires a distance limit between the victim and the attacker since the attacker needs to effectively record the SDR signals emitted from the victim's headset device. According to the official documentation provided by NeuroSky, TGAM is able to transmit within a 10-meter range; however, we found that the signal can be recovered from as far as 22 meters away. In section 7, we examined the performance of signal recovery with respect to the distance constraint. A proximate attack is composed of the following two steps: signal recording and signal recovering.

Signal Recording: Recording the signals emitted from the victim's headset device is the first step for the approximate attack. We used HackRF and GUN Radio to accomplish this step. As described in section 4, the center frequency of the TGAM SDR wave is located approximately at 2458.4 MHz; however, we should avoid placing our center frequency for recording at the same location since signal recording devices such as HackRF generates the so-called Direct Current (DC) offset in its electronics when transmitting or receiving, which interferes with our target SDR. Therefore, in order to circumvent the DC offset which is a signal noise generated by HackRF and

```
                public TG_GetValue
TG_GetValue     proc near               ; DATA XREF

arg_0           = dword ptr  8
arg_4           = dword ptr  0Ch

                push    ebp
                mov     ebp, esp
                mov     eax, [ebp+arg_0]
                mov     ecx, dword_10019240[eax*4]
                test    ecx, ecx
                jz      short loc_10001D22
                mov     eax, [ebp+arg_4]
                cmp     eax, 31h
                ja      short loc_10001D1A
                mov     byte ptr [ecx+eax+1F4h], 0
                fld     dword ptr [ecx+eax*4+12Ch]
                pop     ebp
                retn
```

Fig. 3. The binary codes of `TG_GetValue` in `thinkgear.dll` of a successful data return. As one can see, the `dataType` parameter is passed to `EAX` and the returned value is passed to `st0`.

can cause signal distortion, we placed our recording center frequency according to the following criteria:

$$F_{rc} \leq F_{sc} - \frac{W_{sc}}{2} \text{ or } F_{rc} \geq F_{sc} + \frac{W_{sc}}{2} \tag{1}$$

where F_{rc} is the recording center frequency of HackRF, F_{sc} is the SDR center frequency of the NeuroSky headset which is 2458.4 MHz, and W_{sc} is the SDR occupied bandwidth which is 1 MHz. Therefore, our recording center frequency has to be less than or equal to 2457.9 MHz, or larger than or equal to 2458.9 MHz. We tried the center frequency settings of 2457.9 MHz and 2458.9 MHz, and obtained very similar results. As for the sample rate, we intended to cover at least twice of the useful bandwidth; therefore, we set it to 4 MHz. Having determined these key parameters, we implemented the recording step with GNU Radio which then generates Python scripts for signal collection. The flowgraphs are shown in figure 6a.

Signal Recovering: In the first step, we recorded the victim's TGAM SDR waves and saved them into a raw wave file. In this step, we need to recover the wave file and obtain the EEG data. As mentioned in the first step, in order to avoid the error generated by the DC offset, we set the recording frequency to be 2457.9 or 2458.4 MHz; then in this step we placed the center frequency back to 2458.4 MHz by applying a frequency translating finite impulse response filter with an offset of 500 KHz, which translates the frequency center to 2458.4 MHz. Later

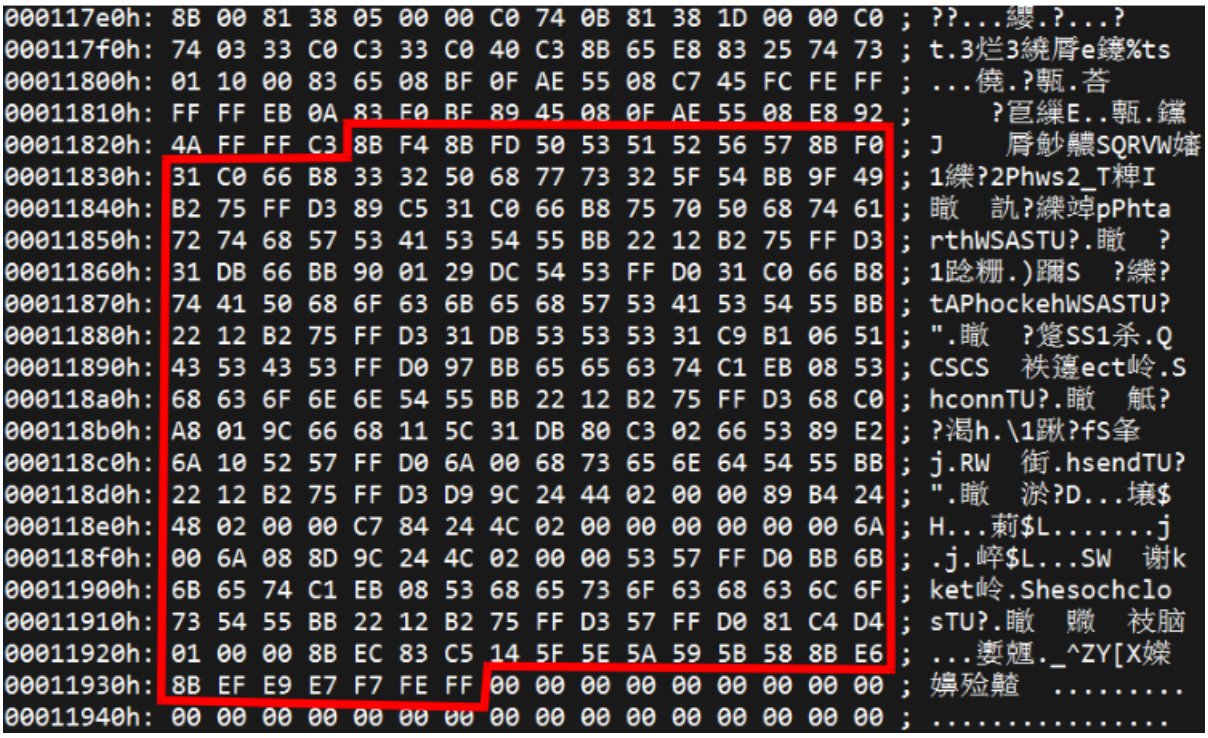


Fig. 4. The ending part of `.text` of `thinkgear.dll`; the red-bordered region includes the injected shellcode.

we further reduced the noises by applying a low pass filter with the sample rate of 160 KHz. We determined this value by gradually shrinking the sample rate and monitoring if the data can be successfully recovered. As a result, 160 KHz turns out to be an effective threshold point. As mentioned in section 4, TGAM leverages MSK for modulation, which is in fact a combination of quadrature modulation and Mueller and Müller (M&M) clock recovery. However, after applying the MSK demodulation, we failed to parse the raw packets based on the format mentioned in the NeuroSky documentation [23]. This can be caused by either a result of an encryption or a packet encoding method other than the documented one. Normally, in order to clarify the true reason, one needs to analyze the firmware of either the TGAM or the RF dongle. Since the firmwares of TGAM and the RF dongle are not publicly available for download, one needs to dump the firmware from the SPI flash on hardware with debugging tools such as JTAG. However, we noticed that the SDR waves are roughly repeating periodically. Hence, we assumed that TGAM does not use any advanced encryption scheme such as AES, to encrypt the waves. In order to confirm this assumption, we employed the entropy analysis method proposed in [39] to analyze the encryption strength. We collected 10 samples of the SDR waves emitted by the headset for approximately 1 minute each, and demodulated the waves with MSK to get 10 raw binary files. For each file, we calculated the Shannon entropy with the following expression:

$$H(j) = -\sum_{i=1}^{n_j} p_j(i) \log_2 p_j(i) \tag{2}$$

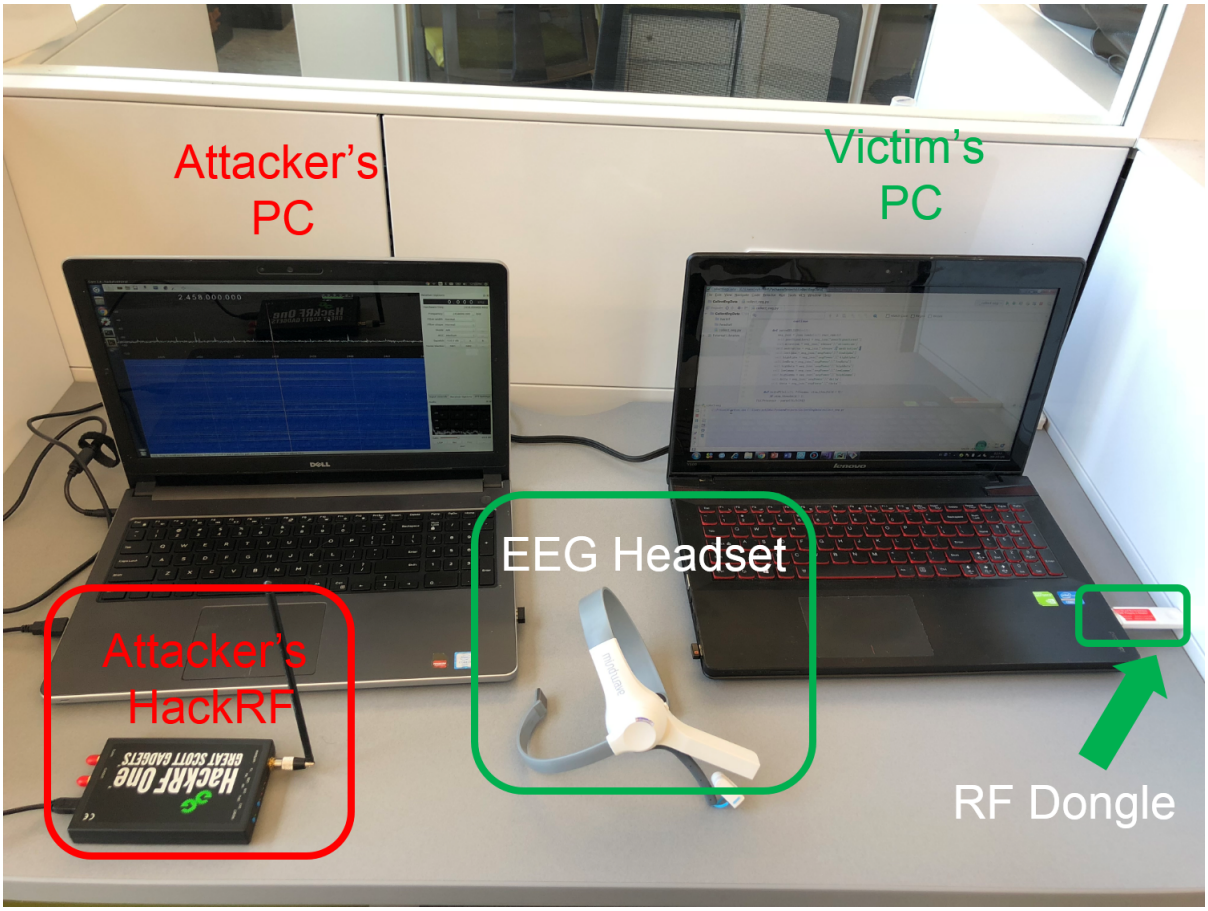
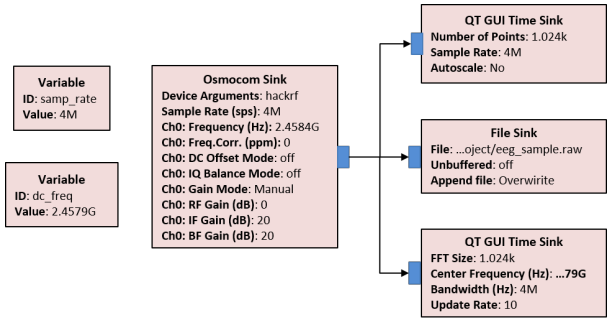


Fig. 5. Configuration of the remote attack. An attacker is on the left side with a HackRF collecting the victim's EEG data emitted from the brain wave headset by means of SDR waves. The right side is the victim's PC with a RF dongle receiving the SDR waves.

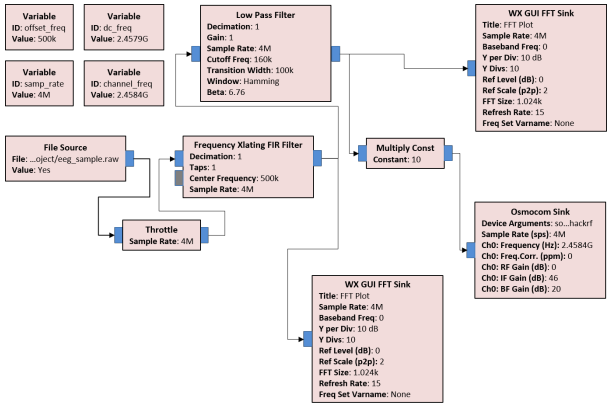
where $H(j)$ is the entropy for the jth binary file, n_j is the total number of unique bytes in the jth binary file, and $p_j(i)$ is the probability of the unique ith byte appearing in the n_j unique bytes of the jth file.

After calculating all the entropies, we took the average of them which yields 5.490510. According to [39], this number falls between the categories of native executables and packed executables. Therefore, it is very likely that the raw waves are trivially encrypted, e.g., XOR ciphering, meaning that simply replaying the waves may recover the EEG data. Therefore, in order to thoroughly prove this assumption, we used one set of devices to record signals and replay the recorded signals in the RF dongle of another set of devices for data recovery. We repeated the experiment for 10 times and the EEG data are all successfully recovered, which proves our assumption. The GNU Radio flowgraph is shown in figure 6b.

In summary, we constructed 4 remote attacks and 1 proximate attack. The two remote attacks serving as malicious BCI apps exploit either the standard SDK or TGSP; the third remote attack serves as a malicious TGSP server; and the last remote attack works as a malicious SDK. In the proximate attack, an attacker first records



(a) Flowgraph for Recording



(b) Flowgraph for Replaying

Fig. 6. GNU Radio flowgraphs for the proximate attack.

the SDR waves emitted from the victim's headset device, and then replays the recorded SDR in his RF dongle to recover the victim's EEG signals. The setting for the proximate attack is shown in figure 5.

6 USER ACTIVITY INFERENCE THROUGH EEG DATA

What information can be reflected from the EEG data remains to be an active research topic. Besides the traditional information such as the focal brain diagnosis, attention level, and meditaion level, as we mentioned above, current research shows that EEG can possibly be used to predict viewed images [14], recognize individuals [26], transfer brain waves to texts [50], monitor sleep [42], and reveal personal information [40]. However, the EEG data used in these research projects were collected by strict medical-use or research-use EEG devices. Whether the reduced-featured EEG data collected by home-use EEG IoT devices also have the potential to reveal user's activities remains unexplored. In this section, we present our deep learning model which can infer a user's current focusing activities with a high accuracy. As one can see, EEG data reflects rich information about a person's health condition; therefore, leakage of the personal EEG data is a severe violation of privacy.

6.1 Overview

Inspired by Zhang *et al.*'s work [34, 50], we leveraged the idea of parallel feature learning and built our own RCNN to classify user's focusing activities. RCNN is composed of a recurrent neural network (RNN) and a convolutional

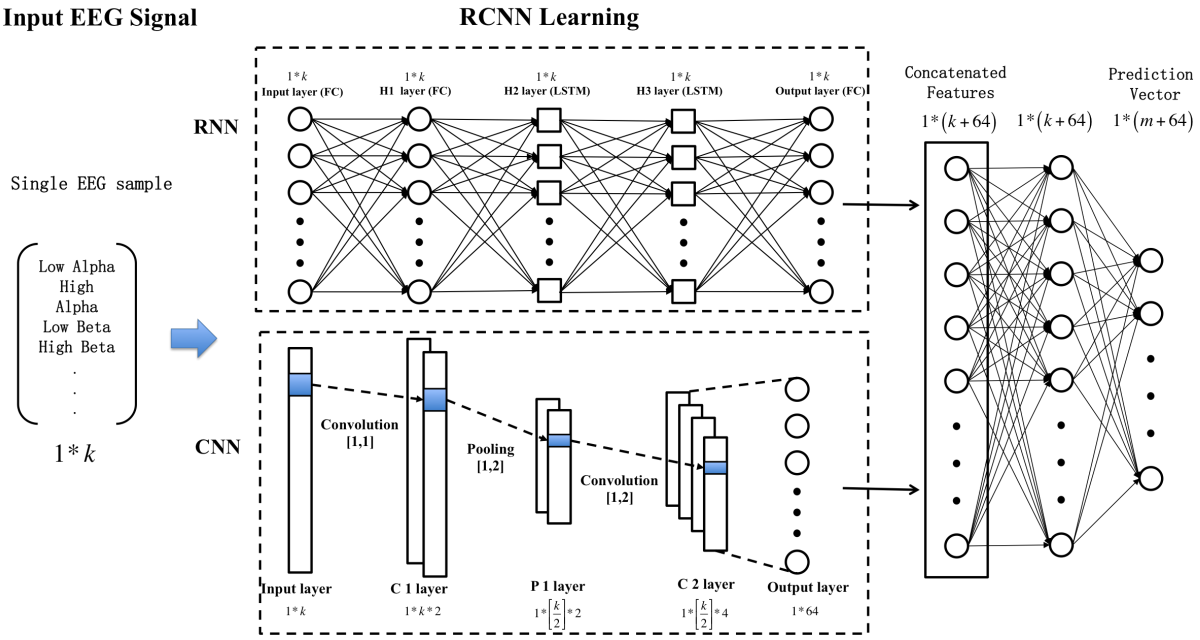


Fig. 7. The RCNN structure. There are 5 layers in the RNN model and 5 layers in the CNN model. The features output by RNN and CNN are concatenated by 3 fully connected layers for the final predictions.

neural network (CNN). The reason for using RNN is because it is able to learn temporal features and EEG data is temporally correlated; and the reason for adopting CNN is because it is capable of learning the potential spatial features and EEG data is spatially correlated as well. Finally, we flattened all the features output by the CNN and RNN networks, concatenated them with several layers of fully connected feedforward neural networks to output the final predictions. The overall structure of the neural network is shown in figure 7. In order to identify a proper structure for our problem, we tested different numbers of layers for both RNN and CNN, from 3 layers to 8 layers; it turned out that a 5-layer structure for both RNN and CNN is the most effective because the in-sample accuracy of our training data increases most rapidly in this structure among all others. Note that we cannot directly apply Zhang *et al.*'s work to our case because: (1) the size of the input features of their model is much higher than that of ours since they used BCI2000 data, (2) the prediction classifications are totally different, and (3) their model contains an autoencoder and an autodecoder which fit for more complicated situations. In the following two subsections we detail the RNN and CNN structures.

6.2 RNN Learning

Our RNN has 5 layers in total: one fully connected feedforward layer as the input layer, three hidden layers which contain one layer of feedforward network and two layers of Long Short-Term Memory (LSTM) cells, and one fully connected feedforward layer as the output layer. Each layer contains k neurons/cells where k is the number of features of the EEG data. In our RNN model, EEG data is first resized into a 2-D vector with a shape of $[n, k]$, where n is the number of samples, or batch size, before the data is fed into the network. For better elaboration, we employ $n = 1$ meaning that only one sample of the features is fed into the network, to present the basic idea. Suppose we denote the data output by the i-th layer of our RNN model to be X_i^r, where $i = 1, 2, \cdots, 5$, the weight

vector between the i-th layer and the $i+1$-th layer to be $W^r_{i,i+1}$, and the bias vector of the i-th layer to be B^r_i. Having presented these notations, we can represent the relationship between the data from two neighboring layers in our model as:

$$X^r_{i+1} = \max(0, X^r_i * W^r_{i,i+1} + B^r_i) \tag{3}$$

where $\max(0, x)$ is the rectifier linear unit (ReLU) used for the activation function. Having had a big picture of our RNN structure, our next step is to take a closer look at each LSTM cell. Suppose at time t, we denote x_t to be a scalar data fed into a LSTM cell, h_t to be the output data by the cell, f_I to be the input gate, f_F to be the forget gate, f_O to be the output gate, and C_t to be the cell state. Within a LSTM cell, the following calculations are performed:

$$f_I = \sigma(W_I \cdot h_{t-1} + W_I \cdot x_t) + B_I \tag{4}$$

$$\widetilde{C}_t = \tanh(W_C \cdot h_{t-1} + W_C \cdot x_t + B_C) \tag{5}$$

$$f_F = \sigma(W_F \cdot h_{t-1} + W_F \cdot x_t) + B_F \tag{6}$$

$$C_t = f_f * C_{t-1} + f_i * \widetilde{C}_t \tag{7}$$

$$f_O = \sigma(W_O \cdot h_{t-1} + W_O \cdot x_t) + B_O \tag{8}$$

$$h_t = f_o * \tanh(C_t) \tag{9}$$

where $\widetilde{C}_t$ is an intermediate value between C_t and C_{t-1}, the variables with letters W and B refer to the weights and biases, respectively, for that specific gate or cell state, and σ represents the sigmoid function which has the form of

$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} \tag{10}$$

At last, our RNN outputs the features with the shape $[1, k]$.

6.3 CNN Learning

Our CNN has 5 layers as well: one convolutional layer as the input layer, a max pooling layer, two convolutional layers, and one fully connected feedforward layer as the output layer. We denote the data output by the i-th layer to be X^c_i, where $i = 1, 2, \cdots, 5$, the weight vector between the i-th layer and the $i+1$-th layer to be $W^c_{i,i+1}$, and the bias vector of the i-th layer to be B^c_i. Similar to RNN, EEG data is first resized into a 2-D vector with the shape of $[n, k]$. In the first convolutional layer, we set the filter with a size of $[1, 1]$ with zero-padding and output one more dimension in depth. Therefore, X^c_1 has the shape of $[1, k, 2]$. For the max pooling layer, we set the stride to be the shape of $[1, 2]$; therefore, X^c_2 has the shape of $[1, \lceil k/2 \rceil, 2]$. We then set the filters for the following two convolutional layers to be $[1, 2]$ and $[1, 4]$, which yield the shapes of X^c_3 and X^c_4 to be $[1, \lceil k/2 \rceil, 4]$ and $[1, \lceil k/2 \rceil, 4]$, respectively. Then we flatten X^c_4 to feed into the last fully connected feedforward layer which outputs the features with a shape of $[1, 64]$. All layers use the ReLU as the activation function.

6.4 Concatenation Layers

Finally, we need to concatenate the features output by RNN and CNN to generate predictions. Our RNN model outputs features with a shape of $[1, k]$ and the CNN outputs features with a shape of $[1, 64]$. We simply concatenate the features to get the shape of $[1, k+64]$ and feed them to two fully connected feedforward layers which output the features with shapes of $[1, k+64]$ and $[1, m]$, respectively, where m is the number of classes of the user's activities. Both layers use ReLU as the activation function. Lastly, we use the Adam optimization algorithm [36]

to reduce our loss function constructed by the softmax cross entropy shown below:

$$loss = -\sum_{j=1}^{m} y'_j \cdot \log(\frac{e^{y_j}}{\sum_{l=1}^{m} e^{y_l}}) \tag{11}$$

where y_j is the predicted probability for the j-th class and y'_j is the ground-truth probability for the j-th class.

7 EVALUATION

In this section, we evaluate the performances of our designs from three aspects: influence, robustness, and effectiveness. With respect to influence, we conducted analysis on how many apps on the NeuroSky App Store are vulnerable to our attacks. As for robustness, we probed into the circumstances of how effective our attacks are based on different attacking environments. Lastly, we demonstrated the effectiveness by showing the accuracy of our inference attack and comparing the results with other well-known and widely-used machine learning algorithms.

7.1 Influence of the Attacks

In this subsection, we study the numbers of apps in the NeuroSky App store that are affected by our attacks. So far, the NeuroSky App store contains 156 apps, with 31 free apps and 125 non-free apps. The prices of the non-free apps range from $0.99 to $1,495. Since each BCI app has to go through the over-the-air (OTA) transmission protocol which is the only way for communications between a brain wave headset and an RF dongle as we discussed in section 4, all the 156 apps publicly available in the NeuroSky App store are vulnerable to our proximate attack, which exploits the coarse-grained implementation of SDR.

We downloaded all the free apps in the NeuroSky App Store, with 31 in total, and conducted an empirical binary analysis against all of them to investigate how they are impacted by our remote attacks. The reason we did not download the non-free apps is because most of them are quite costly. Note that, since the malicious programs for the first two remote attacks presented in section 5 are malicious BCI apps themselves, we excluded them in this evaluation study. Technically speaking, all apps should be at least vulnerable to one of our remote attacks since they either use standard SDK or TGSP, or both, for data retrieval. Therefore, we took a step further to analyze how many apps are vulnerable to the malicious SDK attack, to the malicious TGSP server attack, and to both attacks. We noticed that the binary payloads of all the 31 apps are not encrypted, though some of them are packed for installation; therefore we directly used IDA Pro for static binary analysis. To determine whether an app employs TGSP, we checked if the binary creates a client TCP socket and connects to `localhost` or `127.0.0.1` via the port number 13854; to determine whether an app uses SDK, we checked if the binary invokes API such as `TG_Connect` and `TG_GetValue`. As we expected, all 31 apps are vulnerable to at least one of our remote attacks. Table 3 reports our analysis results, with 16 apps (51.613%) vulnerable only to the malicious TGSP server attack, 5 apps (16.129%) vulnerable only to the malicious SDK attack, and 10 apps (32.258%) vulnerable to both attacks.

Table 3. The number of free apps vulnerable to the remote attacks.

	# of Vulnerable Apps
Malicious TGSP Server	16 (51.613%)
Malicious SDK	5 (16.129%)
Both Attacks	10 (32.258%)

7.2 Robustness of the Attacks

Since our remote attacks infect user's PC side only, there is no data loss or data corruption compared to the ground-truth. The only risk lies in that our malicious programs may be detected and prevented from execution by anti-virus software or firewall. Hence, we used VirusTotal, which is one of the most powerful virus scan engines integrating 58 antivirus software for detection, and Qihoo 360 antivirus software, which has more than 400 million users around the world, to test our malicious programs. We implemented all the four malicious programs for the remote attacks: two malicious BCI apps, a malicious TGSP server, and a malicious SDK. Both VirusTotal and Qihoo 360 fail to recognize any of our programs as malicious. Hence, our remote attacks are highly insidious.

Based on intuition, one can conclude that the effect of proximate attack should be seriously affected by distance and barriers between the attacker and the victim. Therefore, we conducted two sets of experiments to study how distance and barriers can impact on our proximate attack: with one set in an open area without barriers and one set within three small neighboring faculty offices separating by two walls. We detail the procures in the following two paragraphs.

In order to study how distance can affect the data being recovered, we conducted an experiment in which the attacker is gradually moving away from the victim in an open space. We simultaneously measured the SDR signals emitted by the victim's headset whenever the distance from the victim to the attacker is increased by 2 meters, at both the attacker's side using HackRF and the victim's side using his RF dongle. Each trial process (at one location) lasted approximately 2 minutes. We conducted 12 experimental trials (from 0 to 22 meters away from the victim), yielding roughly 120 pieces of EEG data in average for each trial (ranging from 106 pieces to 134 pieces of data for each trial). Then we replayed the SDR waves at the attacker's RF dongle to recover the EEG data and compared it with the ground-truth EEG data collected at the victim side. Note that no corrupted data can be captured since the EEG system framework employs checksum in the raw packets, which drops all corrupted data – as long as a piece of EEG data is able to be recovered, it is not corrupted. Hence, simply counting the number of recovered data is sufficient to demonstrate the robustness of the proximate attack. We found that after the distance between the attacker and the victim increases to 8 meters, the number of recovered data drops dramatically. When the distance increases to 22 meters, only two pieces of data are recovered. The percentages of the data recovered *vs.* distances is detailed in figure 8.

To study the impact of barriers, we conducted experiments within three small neighboring offices separated by two walls, which mimics a real-world environment in which the attacker and the victim are separated by a few walls. This simulation setting corresponds with our threat model since an adversary of the proximate attack could be a malicious neighbor who is a few walls away from the victim. We conducted 6 trials, with 3 of them for the case in which the attacker and the victim are separated by one wall (their distance is about 1 meter), and the other three for the case when they are separated by two walls (their distance is about 3 meters). Such short distances between the victim and the attacker are considered because we intend to consider the impact of barriers only – as indicated in figure 8, a short distance of 3 meters does not contribute significantly to the loss of packets. Similar to the distance experiments, each trial lasted around 2 minutes, yielding a range of 112 to 128 pieces of data for each trial. By repeating the same process we did for EEG data recovery in the open area, we found that in average about 84.09% of data can be recovered if there is one wall as a barrier, and about 47.39% of data can be recovered if there are two walls as barriers.

7.3 Effectiveness of the Inference Attack

In order to evaluate our RCNN learning model, we need to have data for training and testing. Gathering data by ourselves is a tough and time-consumming task which involves recruiting volunteering, puchasing devices, setting up testing environments, and so on. Fortunately, NeuroSky published a large dataset publicly available on Kaggle for research [16], which contains 9,959 pieces of data collected by 30 volunteers. Each volunteer wore a

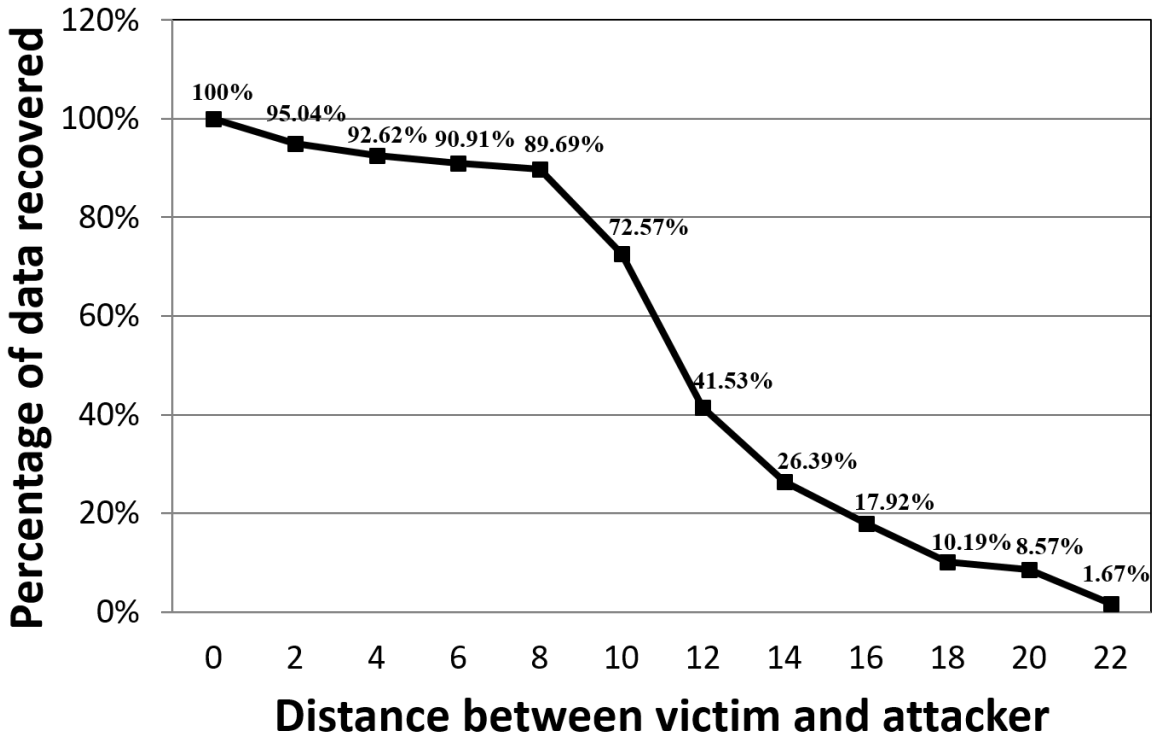


Fig. 8. The percentage graph of the EEG data being recovered *vs.* the distance between the attacker and the victim. The y-axis is the percentage of data being recovered over the ground-truth data, while the x-axis is the distance in meters.

NeuroSky headset constantly collecting the participant's EEG data for different tasks. Each data record contains a piece of reduced-featured EEG data with 11 features: signal poorness level, attention meter, meditation meter, and the 8 types of EEG waveforms as described in section 3. All these 11 features can be obtained by an attacker through our attacks, either from raw-packet format or JSON format as demonstrated in section 4. There are 9 activities a participant may work on: reading instructions, blinking, seeing colors, solving math problems, listening to music, getting ready for next task, relaxing, thinking of an item, and watching videos. Each piece of data completely defines an activity. Echo back to our RCNN model, we have $n = 9959$, $k = 11$ and $m = 9$.

To evaluate the effectiveness of our model, we compared the prediction results by our model with those of other 11 most widely-used and well-known machine learning classifiers: SVM, Guassian Bayes, Bernoulli Bayes, Multinomial Bayes, K-nearest neighbors (KNN), decision tree, random forest, multi-layer perceptron (MLP), adaboost, quadratic discriminant analysis, and logistic regression. In a traditional statistical context, cross-validation is widely adopted for verifying the effectiveness of a learning model. However, in deep learning, it is a known fact that cross-validation should be avoided since it is mostly suitable for small-sized datasets [13]. In each simulation trial, we randomly chose 1,000 pieces of data for testing and used the rest for training, and applied each of the 12 algorithms for prediction; we repeated this procedure for 50 times and calculated the average prediction accuracies for reach classification algorithm. The results are reported in table 4. One can see that our RCNN model has an average accuracy of 70.55%, which far exceeds those of all other popular classifiers.

Table 4. Inference accuracies of different classifiers.

	Accuracy
Random Guess	11.11%
SVM	29.59%
KNN	10.35%
Gaussian Bayes	25.92%
Bernoulli Bayes	24.44%
Multinomial Bayes	10.17%
Decision Tree	24.94%
Random Forest	30.72%
MLP	17.41%
Adaboost	28.79%
Quadratic Discriminant	24.14%
Logistic Regression	27.61%
Our RCNN Model	70.55%

Note that we conducted the simulation study when 10% and 20% of the test data were dropped and obtained exactly the same results. This is reasonable as i) the training data is not affected as an adversary can collect training data from all possible channels and ii) each piece of data completely defines one activity.

8 DEFENSE SOLUTIONS

In this section, we demonstrate defense solutions to mitigate our proposed attacks. For mitigating the remote attacks, we propose an OAuth-based framework to mitigate unauthorized applications exploting standard SDK and TGSP, and a signature-based solution to mitigate malicious SDK and malicious TGSP server. For mitigating the proximate attacks, we propose an encryption-based solution to eliminate possible eavesdropping and replay attacks.

8.1 Solutions to Mitigate Remote Attacks

8.1.1 OAuth-based Framework. The root cause of EEG data leakage to malicious applications through SDK and TGSP is because the ThinkGear framework lacks necessary authentication/authorization. Having noticed the cause, we realize that OAuth [33], a well-known authorization and authentication protocol, can be applied to the ThinkGear framework in order to eliminate attacks resulting from unauthorized malicious applications.

We demonstrate in details how OAuth 2.0 implicit granting protocol can be applied to ThinkGear framework. To begin with, NeuroSky has to maintain a service-providing server on which all BCI-app developers have to register their BCI apps, and each BCI app is assigned with an app ID and an app secret (normally a random hash given by the service-providing server). If a BCI app wishes to access a user's EEG data through TGSP or SDK, the TGSP server or `thinkgear.dll` then redirects the user to the service-providing server along with the requesting app ID and app secret. In this case, the service-providing server first checks if the requesting app ID and app secret match as the ones the developer registered. If they are not matched with any records, the request will be denied directly. If they are matched with one of the record, the user will be asked by the service-providing server whether or not the user wishes to grant the permission to the app to access his EEG data. If granted, the service-providing server passes an access token back to the TGSP server or the `thinkgear.dll`, and the TGSP server or the `thinkgear.dll` further passes the token to the BCI app. Then the BCI app can authenticate itself

with the TGSP server or `thinkgear.dll` for EEG data using the access token. The overview of the protocol is shown as figure xxx.

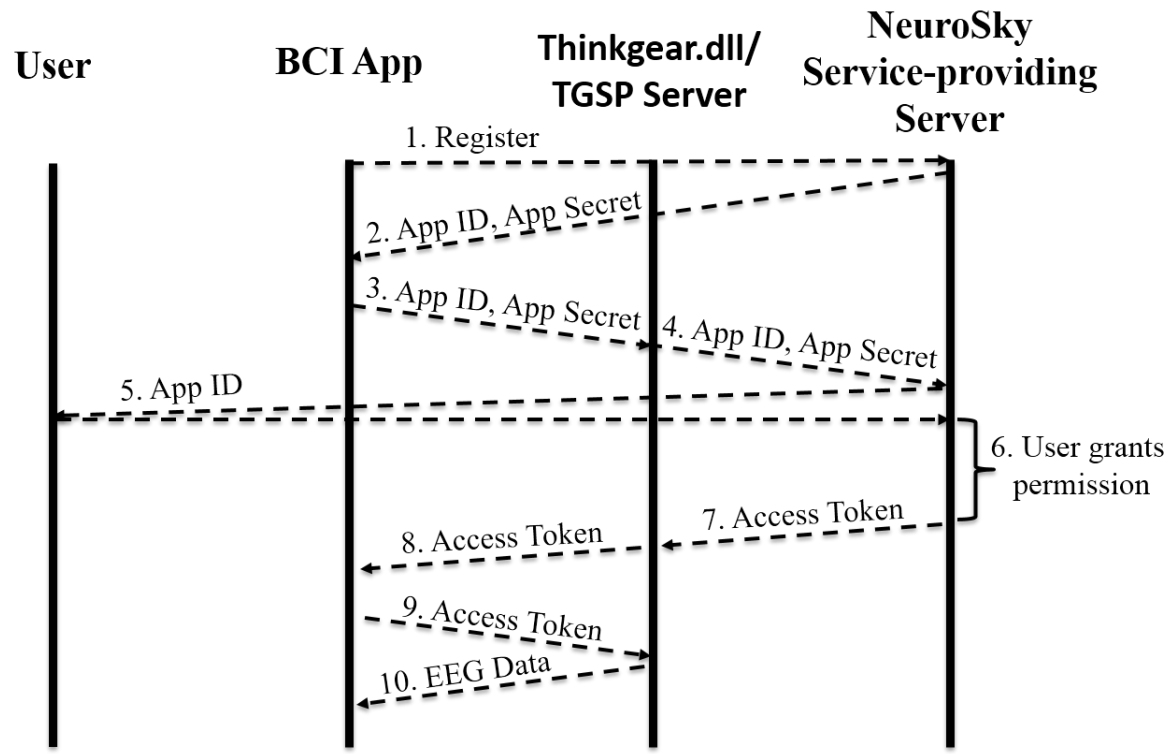


Fig. 9. The OAuth 2.0 implicit granting protocol applied to ThinkGear framework.

8.1.2 Signature-based Solution. NeuroSky does not implement any tamper-resistent techniques, allowing an attacker to arbitrarily modify or fabricate its binary programs, i.e., `thinkgear.dll` and TGSP server. Digital signature is a widely-used mechanism for protecting applications from unauthorized modifications. Digital signature mechanism is consisted with two parts, generation and verification. In the generation step, NeuroSky first needs to generate a hash for each of the protected programs, i.e, `thinkgear.dll` and TGSP server. Then NeuroSky leverages one the asymmetric encryption algorithms by first generating a pair of public key and private key, then using the private key to encrypt the hashes, and finally inserting the encrypted hashes and the digital signature (the public key and the plaintext hashes), into the protected programs. In the verification step, before interacting with `thinkgear.dll` or the TGSP server, every BCI app should first extract the published digital signature and use the public key to decrypt the encrypted hashes, and compare whether the decrypted hashes are matched with the original plaintext hashes. If they are matched, it means the protected programs are intact, otherwise, it means that protected programs have been modified and the BCI apps should terminate the interactions. Digital signature is mature and widely-adopted in the industry, many off-the-shelf tools provide generation and verification services. SignTool is one of the most commonly-used tools developed by Microsoft for this purpose [20].

8.2 Solution to Mitigate Proximate Attack

In order to inhibit the proximate attack, one could eliminate the possibility of replay attack. Simple encryption methods for radio freqency such as voice inversion, hopping inversion and rolling code inversion are easy to be cracked and are vulnerable to replay attacks. However, the computation power of the EEG devices limits the use of some complex encryption protocols, e.g., secure sockets layer (SSL). Having realized this, we suggest adopting advanced encryption standard (AES) for secure EEG data transmission. In order to do so, the NeuroSky headset first generates a random key as a secret key and establishes a Diffie-Hellman (DH) key exchange protocol at a certain channel to share the secret key. Then the headset can transmit the EEG data encrypted with the secret key using AES, then the RF dongle can decrypt the encrypted data with the secret key using AES to obtain the raw EEG data. By adopting the solution properly, NeuroSky can effectively inhibit the proximate attack.

9 CONCLUSION

In this paper, we conducted a thorough security analysis on a typical home-use EEG IoT device, the NeuroSky EEG device. We first demystified the NeuroSky EEG system framework and identified its security weaknesseses via which we implemented two novel easy-to-exploit attack vectors containing four remote attacks and one proximate attack. An attacker can steal a victim's brain wave data via any of the attacks. We further conducted an empirical analysis on the BCI apps in the official NeuroSky App store and found that (i) all 156 apps are vulnerable to our proximate attack, and (ii) all free apps are vulnerable to either the malicious TGSP server attack, or the malicious SDK attack, with 32% of them vulnerable to both attacks. Moreover, we constructed a novel deep learning model to infer user's sensitive activities based on the reduced-featured EEG data collected by the home-use NeuroSky EEG device and obtained an accuracy as high as 70.55%, which far exceeds those of the other 11 well-known and widely-used classifiers we compared against.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Maura Turolla of Telecom Italia for providing specifications about the application scenario.

The work is supported by the National Natural Science Foundation of China under Grant No.: 61273304_a and Young Scientists' Support Program (http://www.nnsf.cn/youngscientists).

REFERENCES

[1] [n. d.]. 2016 Vulnerability Statistics Report. https://www.edgescan.com/assets/docs/reports/2016-edgescan-stats-report.pdf. ([n. d.]). Observed in Oct 2017.
[2] [n. d.]. 2017-2022 Global Wireless EEG Headset Market Analysis. http://www.digitaljournal.com/pr/3439558. ([n. d.]). Observed in Oct 2017.
[3] [n. d.]. 5 Advanced Persistent Threat Trends to Expect in 2016. https://business.f-secure.com/5-advanced-persistent-threat-trends-to-expect-in-2016. ([n. d.]). Observed in Oct 2017.
[4] [n. d.]. 60attacks in 2016. https://www.scmagazineuk.com/60-of-enterprises-were-victims-of-social-engineering-attacks-in-2016/article/576060/. ([n. d.]). Observed in Oct 2017.
[5] [n. d.]. Advanced Persistent Threat. https://en.wikipedia.org/wiki/Advanced_persistent_threat. ([n. d.]). Observed in Oct 2017.
[6] [n. d.]. EEG Devices Market Size Worth $1.40 Billion By 2025. http://www.grandviewresearch.com/press-release/global-electroencephalography-eeg-systems-devices-market. ([n. d.]). Observed in Oct 2017.
[7] [n. d.]. EEG Diagnosis. https://www.healthline.com/health/eeg. ([n. d.]). Observed in Oct 2017.
[8] [n. d.]. EEG: TGAM. https://store.neurosky.com/products/eeg-tgam. ([n. d.]). Observed in Oct 2017.
[9] [n. d.]. GQRX. http://gqrx.dk/. ([n. d.]). Observed in Oct 2017.
[10] [n. d.]. HackRF. http://greatscottgadgets.com/hackrf. ([n. d.]). Observed in Oct 2017.
[11] [n. d.]. Homeland Security warns that certain heart devices can be hacked. http://www.chicagotribune.com/lifestyles/health/ct-cybersecurity-flaw-in-heart-devices-20170111-story.html. ([n. d.]). Observed in Oct 2017.

[12] [n. d.]. IoT Healthcare Market worth 158.07 Billion USD by 2022. http://www.marketsandmarkets.com/PressReleases/iot-healthcare.asp. ([n. d.]). Observed in Oct 2017.
[13] [n. d.]. Is cross-validation heavily used in deep learning or is it too expensive to be used? https://www.quora.com/Is-cross-validation-heavily-used-in-deep-learning-or-is-it-too-expensive-to-be-used. ([n. d.]). Observed in Oct 2017.
[14] [n. d.]. Mind Reading: Using Artificial Neural Nets to Predict Viewed Image Categories From EEG Readings. https://aitopics.org/doc/news:10CDD3C7/. ([n. d.]). Observed in Oct 2017.
[15] [n. d.]. NeuroSky Breaks Guinness World Record to launch the Mindwave headset. https://www.realwire.com/releases/NeuroSky-Breaks-Guinness-World-Record-to-launch-the-Mindwave-headset. ([n. d.]). Observed in Oct 2017.
[16] [n. d.]. NeuroSky EEG Dataset. https://www.kaggle.com/berkeley-biosense/synchronized-brainwave-dataset/data. ([n. d.]). Observed in Oct 2017.
[17] [n. d.]. NeuroSky Headset Test Report. https://apps.fcc.gov/eas/GetApplicationAttachment.html?id=1358958. ([n. d.]). Observed in Oct 2017.
[18] [n. d.]. NeuroSky Introduction. https://en.wikipedia.org/wiki/NeuroSky. ([n. d.]). Observed in Oct 2017.
[19] [n. d.]. Remote Code Execution. https://en.wikipedia.org/wiki/Arbitrary_code_execution. ([n. d.]). Observed in Oct 2017.
[20] [n. d.]. SignTool (Sign Tool). https://docs.microsoft.com/en-us/dotnet/framework/tools/signtool-exe. ([n. d.]). Observed in Jan 2018.
[21] [n. d.]. Smiths Medical Medfusion 4000 Wireless Syringe Infusion Pump Vulnerabilities. https://ics-cert.us-cert.gov/advisories/ICSMA-17-250-02. ([n. d.]). Observed in Oct 2017.
[22] [n. d.]. Social Engineering. https://en.wikipedia.org/wiki/Social_engineering_(security). ([n. d.]). Observed in Oct 2017.
[23] [n. d.]. TGSP Raw Packet. http://developer.neurosky.com/docs/doku.php?id=thinkgear_communications_protocol. ([n. d.]). Observed in Oct 2017.
[24] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. 2017. Understanding the Mirai Botnet. In *USENIX Security Symposium*.
[25] Mirza Mansoor Baig and Hamid Gholamhosseini. 2013. Smart health monitoring systems: an overview of design and modeling. *Journal of medical systems* 37, 2 (2013), 9898.
[26] Lei Chu, Robert Qiu, Haichun Liu, Zenan Ling, and Xin Shi. 2017. Individual Recognition in Schizophrenia using Deep Learning Methods with Random Forest and Voting Classifiers: Insights from Resting State EEG Streams. *arXiv preprint arXiv:1707.03467* (2017).
[27] Andrei Costin, Jonas Zaddach, Aurélien Francillon, Davide Balzarotti, and Sophia Antipolis. 2014. A Large-Scale Analysis of the Security of Embedded Firmwares.. In *USENIX Security Symposium*. 95–110.
[28] Damien Coyle, Jhonatan Garcia, Abdul R Satti, and T Martin McGinnity. 2011. EEG-based continuous control of a game using a 3 channel motor imagery BCI: BCI game. In *Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB), 2011 IEEE Symposium on*. IEEE, 1–7.
[29] Justin Dauwels, François Vialatte, and Andrzej Cichocki. 2010. Diagnosis of Alzheimer's disease from EEG signals: where are we standing? *Current Alzheimer Research* 7, 6 (2010), 487–505.
[30] Simon Eberz, Nicola Paoletti, Marc Roeschlin, Marta Kwiatkowska, I Martinovic, and A Patané. 2017. Broken hearted: How to attack ECG biometrics. In *NDSS Symposium*.
[31] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2016. Security analysis of emerging smart home applications. In *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 636–654.
[32] Flavio D Garcia, David Oswald, Timo Kasper, and Pierre Pavlidès. 2016. Lock It and Still Lose It-on the (In) Security of Automotive Remote Keyless Entry Systems.. In *USENIX Security Symposium*.
[33] Dick Hardt. 2012. The OAuth 2.0 authorization framework. (2012).
[34] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[35] SM Riazul Islam, Daehan Kwak, MD Humaun Kabir, Mahmud Hossain, and Kyung-Sup Kwak. 2015. The internet of things for health care: a comprehensive survey. *IEEE Access* 3 (2015), 678–708.
[36] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[37] G Lantz, R Grave De Peralta, L Spinelli, M Seeck, and CM Michel. 2003. Epileptic source localization with high density EEG: how many electrodes are needed? *Clinical neurophysiology* 114, 1 (2003), 63–69.
[38] Chunxiao Li, Anand Raghunathan, and Niraj K Jha. 2011. Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system. In *e-Health Networking Applications and Services (Healthcom), 2011 13th IEEE International Conference on*. IEEE, 150–156.
[39] Robert Lyda and James Hamrock. 2007. Using entropy analysis to find encrypted and packed malware. *IEEE Security & Privacy* 5, 2 (2007).
[40] Ivan Martinovic, Doug Davies, Mario Frank, Daniele Perito, Tomas Ros, and Dawn Song. 2012. On the Feasibility of Side-channel Attacks with Brain-computer Interfaces. In *Proceedings of the 21st USENIX conference on Security symposium*. USENIX Association.
[41] Jens Müller, Vladislav Mladenov, Juraj Somorovsky, and Jörg Schwenk. 2017. SoK: Exploiting Network Printers. In *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 213–230.

[42] Takashi Nakamura, Valentin Goverdovsky, Mary J Morrell, and Danilo P Mandic. 2017. Automatic sleep monitoring using ear-EEG. *arXiv preprint arXiv:1701.04398* (2017).

[43] Mahmudur Rahman, Bogdan Carbunar, and Madhusudan Banik. 2013. Fit and vulnerable: Attacks and defenses for a health monitoring device. *arXiv preprint arXiv:1304.5672* (2013).

[44] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin OâĂŹFlynn. 2017. IoT goes nuclear: Creating a ZigBee chain reaction. In *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 195–212.

[45] Gerwin Schalk, Dennis J McFarland, Thilo Hinterberger, Niels Birbaumer, and Jonathan R Wolpaw. 2004. BCI2000: a general-purpose brain-computer interface (BCI) system. *IEEE Transactions on biomedical engineering* 51, 6 (2004), 1034–1043.

[46] Agusti Solanas, Constantinos Patsakis, Mauro Conti, Ioannis S Vlachos, Victoria Ramos, Francisco Falcone, Octavian Postolache, Pablo A Pérez-Martínez, Roberto Di Pietro, Despina N Perrea, et al. 2014. Smart health: a context-aware health paradigm within smart cities. *IEEE Communications Magazine* 52, 8 (2014), 74–81.

[47] Kirill Stytsenko, Evaldas Jablonskis, and Cosima Prahm. 2011. Evaluation of consumer EEG device Emotiv EPOC. In *MEi: CogSci Conference 2011, Ljubljana*.

[48] Barbara E Swartz. 1998. The advantages of digital over analog recording techniques. *Electroencephalography and clinical neurophysiology* 106, 2 (1998), 113–117.

[49] Geng Yang, Li Xie, Matti Mäntysalo, Xiaolin Zhou, Zhibo Pang, Li Da Xu, Sharon Kao-Walter, Qiang Chen, and Li-Rong Zheng. 2014. A health-iot platform based on the integration of intelligent packaging, unobtrusive bio-sensor, and intelligent medicine box. *IEEE transactions on industrial informatics* 10, 4 (2014), 2180–2191.

[50] Xiang Zhang, Lina Yao, Quan Z Sheng, Salil S Kanhere, Tao Gu, and Dalin Zhang. 2017. Converting Your Thoughts to Texts: Enabling Brain Typing via Deep Feature Learning of EEG Signals. *arXiv preprint arXiv:1709.08820* (2017).