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

Yinhao Xiao*, Yizhen Jia*, Xiuzhen Cheng*, Jiguo Yu†¶, Zhenkai Liang‡, and Zhi Tian§
*Department of Computer Science, The George Washington University
†Department of Computer Science, Qufu Normal University
‡School of Computing, National University of Singapore
§Department of Computer Science, George Mason University
¶Corresponding Author

***Abstract*—Health-related IoT devices are becoming more popular in recent years. On one hand, users can access information of their health conditions more conveniently; on the other hand, they are exposed to new security risks. 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 reverse-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. 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 real-world EEG data indicates that the inference accuracy of the proposed RCNN is can reach 70.55%.**

I. Introduction

Health-related IoT devices have been constantly attracting public attention in recent years. According to a most recent report, the market for healthcare IoT reaches 41.22 billion USD in 2017 and is projected to grow to 158.07 billion USD by the year of 2022 [1]. 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 popular devices 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 [2]. Hackers can remotely hack into a patient's defibrillator and trigger irregular heart rhythms which can cause a cardiac failure. In September 2017, DHS issued a warning stating that approximately 4,000 wireless syringe infusion pumping devices are vulnerable to remote attacks [3]. Attackers can exploit a loophole to murder a patient by remotely giving the patient overdose infusion. As one can see from these cases, different from compromising traditional IoT devices such as smart hues or thermostats that result 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 number 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 that they have a drastic growing market which is projected to reach 1.40 billion USD by the year of 2025 [4]. 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.* [5] 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 brainwave devices for home use in the world [6], and it still holds the largest market share of home-use EEG devices [7]. 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 brainwave data. 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 [9] [10] and sleep levels [11]. 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 RCNN model 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 real-world 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 proximate attack.

Paper Organization. The rest of the paper is organized as follows. Section II outlines related works. Section III introduces the preliminary knowledge on EEG, and our threat model. Section IV details the demystified EEG system framework. Section V demonstrates the security flaws of the framework and the implementations of our attacks based on these flaws. Section VI presents our deep learning model for the reduced-featured EEG inference attack. Section VII reports the performance of our attacks. Section VIII presents our defense solutions and Section IX concludes the paper.

II. Related Works

In this section, we provide a brief overview of 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 its rapid development and its tight connection to personal healthcare. Eberz *et al.* [12] 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.

Rahman *et al.* [13] investigated the security and privacy issues of Fitbit; after reversely engineering the ANT protocol for data communications, they generated various attacks and proposed the corresponding defense mechanisms. Li *et al.* [14] 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. Martinovic *et al.* [5] explored a research-use EEG device, EPOC, as a potential attack intermediary to infer private information about their users. Compared to the four pieces of research work mentioned above, besides the security of the health-related device itself, we also addressed the problem of sensitive information leak by the captured EEG signals. Eberz *et al.* demonstrated that a user's biometrics, e.g., ECG and eye movements, can be inferred by other fitness data gathered in a different context [15].

Alongside the security study on health-related IoT devices, many works focused on building smart health systems and platforms. In [16] [17], the authors discussed the concept of health-related smart city and the home-based wellness platform. Mirza *et al.* [18] provided an overview of the design and modeling of the current smart health monitoring systems. Coincidentally, ISLAM *et al.* [19] 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 a 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.* [20] 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. They discovered 2 design flaws and developed 5 PoC attacks targetting the SmartThings system. Costin *et al.* [21] 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. Celik *et al.* proposed a novel sensitive information tracking algorithm based on taint analysis to effectively detect the sensitive information leakages in SmartThings market apps [22]. Chen *et al.* presented a fuzzing mechanism which discovers possible memory corruptions in IoT device even the firmware is absent [23].

There also exists research targetting the communication protocols of the IoT systems. For example, Ronan *et al.* [24] discovered a bug in the Touchlink part of the ZigBee Light Link protocol implemented by Philips, and implemented a smart light bulb worm which automatically spreads and controls all the Philips smart lights in a city. Our work is along with a similar line except that we further analyzed the captured EEG signals rather than just taking control of the devices.

III. Background

In this section, we first briefly introduce the taxonomy of EEG devices. Then we demonstrate our threat model for the home-use EEG devices.

A. EEG Taxonomy

According to our research, the current EEG methods and their corresponding devices can be classified into three cate-

gories based on the uses of EEG: for strict medical use, for research use, and for home use. Those for strict medical-use are mainly employed for focal diagnosis. A headset of this type of EEG normally has 125 electrodes and is not publicly available on the market for sale [25]. The headset of research-Use EEG normally has 64 electrodes [26]. Such devices are not available for purchase in the markets either. 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. Its market is dominated by a company called NeuroSky [27], which is the first company to commercialize home-use EEG devices and maintains the largest home-use EEG market share so far [6], [7]. NeuroSky invented the PCB module, i.e., the TGAM, which serves as a multi-mode sensor to collect EEG data, and the EEG system framework for EEG data transmission. It has an App store where users can purchase or download BCI apps. It also provides an 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 of home-use EEG. 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. The attention meter and the meditation meter are both valuing from 0 to 100, indicating the level of the attention and that of the relaxation of a human being, respectively. The techniques to calculate the attention and meditation levels are quite mature, and have been extensively studied in the field of biomedical research [28], [29]. In this paper, we do not intend to cover the corresponding algorithms since they are out of the scope. Besides these two meters, the other 8 types of data are traditional EEG waveforms and can be directly measured through different frequencies illustrated as follows.

- **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 values: 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 values: 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 values: 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 sound and sight.

B. Threat Model

In our adversarial model, attackers aim to steal brain waves of NeuroSky users and infer sensitive personal activitites 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 [30], or exploit the known RCE vulnerabilities [31]. Note that these techniques are all viable and feasible, and have been widely adopted as attack vectors for compromising industrial systems [32] [33] [34]. 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.

IV. Demystifying the EEG System Framework

In this section, we introduced the two sides of the EEG system framework, the user application side where high-level BCI apps reside and the headset side where brain waves are collected and transmitted.

A. 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 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).

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`. We reversely engineered the DLL and found that besides the DLL main entry function `DLLEntryPoint`, it exports 19 other API functions. After a further exploration, we identified 4 functions that are mandatory to complete an

EEG data retrieval, namely `TG_GetNewConnectionId`, `TG_Connect`, `TG_ReadPackets`, and `TG_GetValue`, which are called in the above order; 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.

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.

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 follow the following procedure: TGC first opens a TCP connection with a fixed port number `13854`, then a BCI app registers itself. After the registration process, the BCI app sends a request to retrieve the EEG data from TGC. If requested with the JSON format, TGC responds with the detailed information. After the verification process, the BCI app can send a request to retrieve the EEG data from TGC. If requested with the JSON format, TGC responds with the detailed information.

B. 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). 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. We found that TGAM chip can operate on the frequency band ranging from 2419.9 MHz to 2470.9 MHz by looking up the Federal Communication Comissino (FCC) ID printed on the EEG IoT device. We then discovered the approximate center frequency of 2458.4 MHz with an occupied bandwidth of approximately 1 MHz through HackRF [35] and GQRX [36]. 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 as we mentioned above.

According to our experimental study and analysis described above, we concluded that the entire EEG system framework can be sketched as in figure 1. It consists 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.

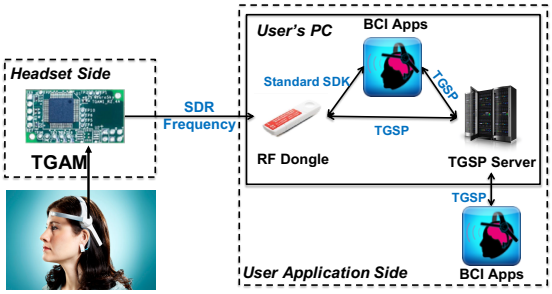

Fig. 1: EEG system framework

V. Implementation of Attack Vectors

In this section, we detail our novel attack vectors targeting the EEG system framework we demystified in Section IV. Our attack vectors can be classified as either remote attacks or proximate attacks.

A. 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.

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 IV, 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. This is done by letting the malicious app call `TG_GetNewConnectionId` and `TG_ConnectTG_ReadPackets` in sequence. Since none of the APIs imposes any authentication restriction for calling, our malicious app steals the EEG data without any authentication or verification. However, if there is another BCI app running, this method fails. We then conducted dynamic debugging on the binary `thinkgear.dll` to

explore the reason 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 at every level, imposing a tremendous burden on 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 IV, 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.

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 IV, the TGSP server adopts a naive registration 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 an 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.

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 IV. This attack is fully implemented in C/C++.

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 IV, 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, 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 2, 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 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` sections are 70,692 bytes and 71,168 bytes, respectively, meaning that we need to squeeze the size of our shellcode to be less than 476 bytes. We then implemented a shellcode with a size of 275 bytes, which can steal the two values and pass to our malicious server via a TCP connection.

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.

B. Proximate Attack

Different from the remote attacks in which a malicious program is required, launching a proximate attack does not require the 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;

```
                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. 2: 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`.

however, we found that the signal can be recovered from as far as 22 meters away. In section VII, 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 proximate attack. We used HackRF and GUN Radio to accomplish this step. As described in section IV, 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 can cause signal distortion, we placed our recording center frequency according to the following criteria:

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

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. 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.

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 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. Then we employed the entropy analysis method proposed in [37] to analyze the encryption strength of raw packets and found that it is highly 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 verify this possibility, 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.

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 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.

VI. 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 [38], recognize individuals [39], transfer brain waves to texts [40], monitor sleep [11], and reveal personal information [5]. 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.

A. Overview

Inspired by Zhang *et al.*'s work [40], [41], 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 neural network (CNN). 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 3. 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 more complicated situations. In the following two subsections we detail the RNN and CNN structures.

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

$$X_{i+1}^r = \max(0, X_i^r * W_{i,i+1}^r + B_i^r) \tag{2}$$

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:

$$f_I = \sigma(W_I \cdot h_{t-1} + W_I \cdot x_t) + B_I \tag{3}$$

$$\widetilde{C}_t = \tanh(W_C \cdot h_{t-1} + W_C \cdot x_t + B_C) \tag{4}$$

$$f_F = \sigma(W_F \cdot h_{t-1} + W_F \cdot x_t) + B_F \tag{5}$$

$$C_t = f_f * C_{t-1} + f_i * \widetilde{C}_t \tag{6}$$

$$f_O = \sigma(W_O \cdot h_{t-1} + W_O \cdot x_t) + B_O \tag{7}$$

$$h_t = f_o * \tanh(C_t) \tag{8}$$

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

$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} \tag{9}$$

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

C. 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_i^c, where $i = 1, 2, \cdots, 5$, the weight vector between the i-th layer and the $i+1$-th layer to be $W_{i,i+1}^c$, and the bias vector of the i-th layer to be B_i^c. 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_1^c 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_2^c 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_3^c and X_4^c to be $[1, \lceil k/2 \rceil, 4]$ and $[1, \lceil k/2 \rceil, 4]$, respectively. Then we flatten X_4^c 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.

D. 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 into 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 to reduce our loss function constructed by the softmax cross entropy [42].

E. Philosophy of RCNN Modeling

Currently, the interpretability of deep learning is still not sufficiently explored because of its high non-linear architecture and black-box structure [43]. Due to this low interpretability, researchers designed their deep learning models starting with a rough intuition and then adjusting their models (e.g., adding layers or neurons) to achieve a sound performance without giving strict theoretical analysis [44], [45]. Similarly, in this paper, we developed our RCNN model based on intuition without theoretically verify whether or not our model is completely suitable for tackling the problem.

The philosophy behind our RCNN modeling is based on the fact that EEG data is khown to possess both spatial and temporal correlations [46], [47]. Thus we perceived to design a model that can seize the spatial and temporal features of EEG. Noticing that CNN has the capability of capturing spatial features while RNN has the potential to capture the temporal features, and that fully-connected layers are able to combine the features output from CNN and RNN, we designed a RCNN model shown in Figure 3 for our inference attack. Note that it

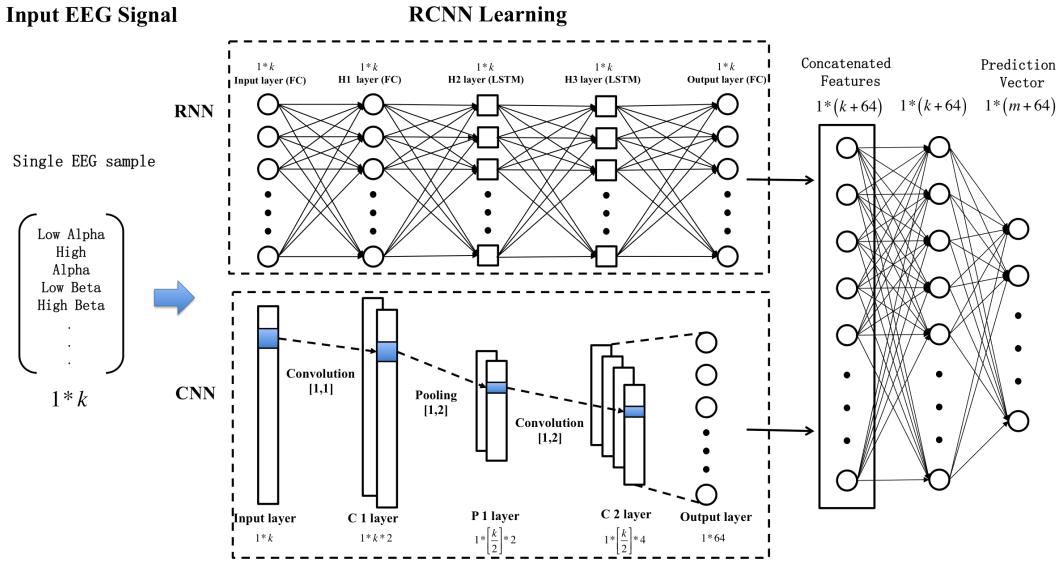

Fig. 3: The RCNN structure. There are 5 layers in the RNN model and 5 layers in the CNN model. The features output by RNN and CNN are concatenated by 3 fully connected layers for the final predictions.

is not feasible to exhaustively explore every possible variation of RCNN. We tested multiple models ranging from two layers to ten layers, and found that our current design has the fastest decrease speed in the lost function.

VII. Evaluation

In this section, we evaluate the performances of our designs from three aspects: influence, robustness, and effectiveness.

A. 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. Since each BCI app has to go through the over-the-air (OTA) transmission protocol which is the only way for communications between a brainwave headset and an RF dongle as we discussed in section IV, 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 31 free apps in the NeuroSky App Store and conducted an empirical binary analysis against all of them. Note that, since the malicious programs for the first two remote attacks presented in section V are malicious BCI apps themselves, we excluded them in this evaluation study. Then we analyzed 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 I reports our analysis results.

TABLE I: 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%)

B. Robustness of the Attacks

Since our remote attacks infect a 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. Both VirusTotal and Qihoo 360 fail to recognize any of our programs as malicious. Hence, our remote attacks are highly insidious.

We then 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 separated by two walls. We detailed the procedures in the following two paragraphs.

In order to study how distance can affect the data being recovered, 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. 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). We found that after the distance between the attacker and the victim increases to 8 meters, the number of recovered data drops dramatically. The percentages of the data recovered *vs.* distances is detailed in Figure 4.

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. 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). 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.

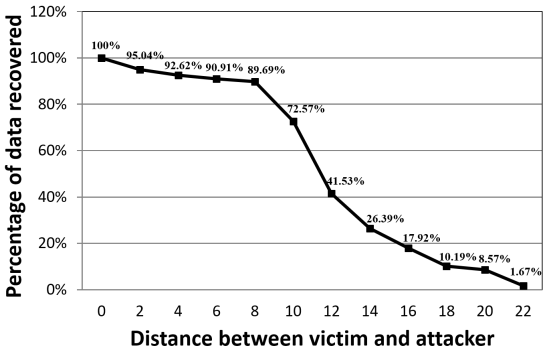

Fig. 4: The percentage graph of the EEG data being recovered *vs.* the distance between the attacker and the victim. The y-axis is the percentage of data being recovered over the ground-truth data, while the x-axis is the distance in meters.

C. Effectiveness of the User Activity Inference

In order to evaluate our RCNN learning model, we need to have data for training and testing. Fortunately, NeuroSky published a large dataset publicly available on Kaggle for research [48], which contains 9,959 pieces of data collected by 30 volunteers. Each volunteer wore a NeuroSky headset constantly collecting the participant's EEG data for different tasks. Each data record contains a piece of reduced-featured EEG data with 10 features as described in section III plus the signal poorness level. 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 $m = 9$.

TABLE II: 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%

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, Gaussian 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 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 accuracy for reach classification algorithm. The averaged prediction accuracy is defined as follows:

$$\frac{1}{50}\sum_{i=1}^{50}\frac{|\{y_{ij}|y_{ij}=y'_{ij},\forall 1\leq j\leq 1000\}|}{1000} \tag{10}$$

where i represents the i-th prediction trial (totally 50 trials), j represents the j-th piece of the test data (totally 1,000 pieces of data), y_{ij} is the predicted result of the j-th testing data at the i-th trial, and y'_{ij} is the ground truth label of the j-th testing data at the i-th trial.

Our RCNN model has an average accuracy of 70.55%, which far exceeds those of all other popular classifiers whose largest accuracy value is 30.72%. 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. The averaged prediction accuracies of all classifiers are shown in Table II.

VIII. 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 exploiting 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.

A. Solutions to Mitigate Remote Attacks

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 [49], a well-known authorization and authentication protocol, can be applied to the ThinkGear framework in order to eliminate attacks resulting from unauthorized malicious applications.

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 can be applied to protect applications from unauthorized modifications.

B. Solution to Mitigate Proximate Attack

In order to inhibit the proximate attack, one could eliminate the possibility of the replay attack. Simple encryption methods for radio frequency 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 complex encryption protocols such as the secure sockets layer (SSL). Having realized this, we suggest adopting advanced encryption standard (AES) for secure EEG data transmissions.

IX. 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 weaknesses 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 brainwave 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.

References

[1] "Iot healthcare market worth 158.07b usd," http://www.marketsandmarkets.com/PressReleases/iot-healthcare.asp, 2018.

[2] "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, 2018.

[3] "Smiths medical medfusion 4000 wireless syringe infusion pump vulnerabilities," https://ics-cert.us-cert.gov/advisories, 2018.

[4] "Eeg market size worth $1.40 billion by 2025," http://www.grandviewresearch.com/press-release/global-electroencephalography-eeg-systems-devices-market, 2018.

[5] I. Martinovic, D. Davies, M. Frank, D. Perito, T. Ros, and D. Song, "On the feasibility of side-channel attacks with brain-computer interfaces," in *Proceedings of the 21st USENIX conference on Security symposium*. USENIX Association, 2012.

[6] "Neurosky breaks guinness world record to launch the mindwave headset," https://www.realwire.com/releases, 2018.

[7] "2017-2022 global wireless eeg headset market analysis," http://www.digitaljournal.com/pr/3439558, 2018.

[8] "Eeg: Tgam," https://store.neurosky.com/products/eeg-tgam, 2018.

[9] "Eeg diagnosis," https://www.healthline.com/health/eeg, 2018.

[10] J. Dauwels, F. Vialatte, and A. Cichocki, "Diagnosis of alzheimer's disease from eeg signals: where are we standing?" *Current Alzheimer Research*, vol. 7, no. 6, pp. 487–505, 2010.

[11] T. Nakamura, V. Goverdovsky, M. J. Morrell, and D. P. Mandic, "Automatic sleep monitoring using ear-eeg," *arXiv:1701.04398*, 2017.

[12] S. Eberz, N. Paoletti, M. Roeschlin, M. Kwiatkowska, I. Martinovic, and A. Patané, "Broken hearted: How to attack ecg biometrics," in *NDSS Symposium*, 2017.

[13] M. Rahman, B. Carbunar, and M. Banik, "Fit and vulnerable: Attacks and defenses for a health monitoring device," *arXiv:1304.5672*, 2013.

[14] C. Li, A. Raghunathan, and N. K. Jha, "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, 2011, pp. 150–156.

[15] S. Eberz, G. Lovisotto, A. Patanè, M. Kwiatkowska, V. Lenders, and I. Martinovic, "When your fitness tracker betrays you: Quantifying the predictability of biometric features across contexts," in *2018 IEEE Symposium on Security and Privacy (SP)*, May 2018, pp. 889–905.

[16] A. Solanas, C. Patsakis, M. Conti, I. S. Vlachos, V. Ramos, F. Falcone, O. Postolache, P. A. Pérez-Martínez, R. Di Pietro, D. N. Perrea *et al.*, "Smart health: a context-aware health paradigm within smart cities," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 74–81, 2014.

[17] G. Yang, L. Xie, M. Mäntysalo, X. Zhou, Z. Pang, L. Da Xu, S. Kao-Walter, Q. Chen, and L.-R. Zheng, "A health-iot platform based on the integration of intelligent packaging, unobtrusive bio-sensor, and intelligent medicine box," *IEEE transactions on industrial informatics*, vol. 10, no. 4, pp. 2180–2191, 2014.

[18] M. M. Baig and H. Gholamhosseini, "Smart health monitoring systems: an overview of design and modeling," *Journal of medical systems*, vol. 37, no. 2, p. 9898, 2013.

[19] S. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K.-S. Kwak, "The internet of things for health care: a comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, 2015.

[20] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 636–654.

[21] A. Costin, J. Zaddach, A. Francillon, D. Balzarotti, and S. Antipolis, "A large-scale analysis of the security of embedded firmwares." in *USENIX Security Symposium*, 2014, pp. 95–110.

[22] Z. B. Celik, L. Babun, A. K. Sikder, H. Aksu, G. Tan, P. McDaniel, and A. S. Uluagac, "Sensitive information tracking in commodity iot," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, 2018, pp. 1687–1704. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/celik

[23] J. Chen, W. Diao, Q. Zhao, C. Zuo, Z. Lin, X. Wang, W. Lau, M. Sun, R. Yang, and K. Zhang, "Iotfuzzer: Discovering memory corruptions in iot through app-based fuzzing," in *2018 NDSS*, 01 2018.

[24] E. Ronen, A. Shamir, A.-O. Weingarten, and C. O'Flynn, "Iot goes nuclear: Creating a zigbee chain reaction," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 195–212.

[25] G. Lantz, R. G. De Peralta, L. Spinelli, M. Seeck, and C. Michel, "Epileptic source localization with high density eeg: how many electrodes are needed?" *Clinical neurophysiology*, vol. 114, pp. 63–69, 2003.

[26] G. Schalk, D. J. McFarland, T. Hinterberger, N. Birbaumer, and J. R. Wolpaw, "Bci2000: a general-purpose brain-computer interface (bci) system," *IEEE Transactions on biomedical engineering*, vol. 51, no. 6, pp. 1034–1043, 2004.

[27] "Neurosky introduction," https://en.wikipedia.org/wiki/NeuroSky, 2018.

[28] L. Aftanas and S. Golocheikine, "Human anterior and frontal midline theta and lower alpha reflect emotionally positive state and internalized attention: high-resolution eeg investigation of meditation," *Neuroscience letters*, vol. 310, no. 1, pp. 57–60, 2001.
[29] T. Takahashi, T. Murata, T. Hamada, M. Omori, H. Kosaka, M. Kikuchi, H. Yoshida, and Y. Wada, "Changes in eeg and autonomic nervous activity during meditation and their association with personality traits," *International Journal of Psychophysiology*, vol. 55, no. 2, pp. 199–207, 2005.
[30] A. KOYUN and E. Al Janabi, "Social engineering attacks," *Multidisciplinary Engineering Science and Technology*, 2017.
[31] K. Kaspersky and A. Chang, "Remote code execution through intel cpu bugs," in *Hack In The Box (HITB) 2008 Malaysia Conference*, 2008.
[32] "60% of enterprises were victims of social engineering attacks in 2016," https://www.scmagazineuk.com, 2018.
[33] "5 advanced persistent threat trends to expect in 2016," https://business.f-secure.com, 2018.
[34] "2016 vulnerability statistics report," https://www.edgescan.com/assets/docs/reports/2016-edgescan-stats-report.pdf, 2018.
[35] "Hackrf," http://greatscottgadgets.com/hackrf, 2018.
[36] "Gqrx," http://gqrx.dk/, 2018.
[37] R. Lyda and J. Hamrock, "Using entropy analysis to find encrypted and packed malware," *IEEE Security & Privacy*, vol. 5, no. 2, 2007.
[38] "Mind reading: Using artificial neural nets to predict viewed image categories from eeg readings," https://aitopics.org/doc, 2018.
[39] L. Chu, R. Qiu, H. Liu, Z. Ling, and X. Shi, "Individual recognition in schizophrenia using deep learning methods with random forest and voting classifiers: Insights from resting state eeg streams," *arXiv:1707.03467*, 2017.
[40] X. Zhang, L. Yao, Q. Z. Sheng, S. S. Kanhere, T. Gu, and D. Zhang, "Converting your thoughts to texts: Enabling brain typing via deep feature learning of eeg signals," *arXiv:1709.08820*, 2017.
[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[42] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.
[43] W. Samek, T. Wiegand, and K. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," *CoRR*, vol. abs/1708.08296, 2017. [Online]. Available: http://arxiv.org/abs/1708.08296
[44] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 815–823.
[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.
[46] P. F. Chang, L. Arendt-Nielsen, and A. C. Chen, "Dynamic changes and spatial correlation of eeg activities during cold pressor test in man," *Brain research bulletin*, vol. 57, no. 5, pp. 667–675, 2002.
[47] K. Linkenkaer-Hansen, V. V. Nikouline, J. M. Palva, and R. J. Ilmoniemi, "Long-range temporal correlations and scaling behavior in human brain oscillations," *Journal of Neuroscience*, vol. 21, no. 4, pp. 1370–1377, 2001.
[48] "Neurosky eeg dataset," https://www.kaggle.com/berkeley-biosense/synchronized-brainwave-dataset/data, 2018.
[49] D. Hardt, "The oauth 2.0 authorization framework," *RFC*, 2012.