Network Forensics of Ransomware and Sensor Integrity in Machine Learning; Toward Building Robust Systems against Cyberattacks

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2024

Ade KURNIAWAN

List of publication

Journal papers

1. A. Kurniawan and I. Riadi, "Detection and Analysis Cerber Ransomware Using Network Forensics Behavior Based," *International Journal Network and Security*, vol. 20, no. 5, pp. 1–8, 2018, doi: 10.6633/IJNS.201809_20(5).04.

2. A. Kurniawan, Y. Ohsita, and M. Murata, "Experiments on Adversarial Examples for Deep Learning Model Using Multimodal Sensors," *Sensors*, vol. 22, no. 22, p. 8642, Nov. 2022, doi: 10.3390/s22228642.

Refereed Conference Papers

1. Ade Kurniawan, Yuichi Ohsita, and Masayuki Murata, "Toward Robust Systems against Sensor-Based Adversarial Examples Based on the Criticalities of Sensors," to be presented at IEEE 3rd International Conference on AI in Cybersecurity (ICAIC), Houston, Texas, February 7–9, 2024.

Preface

As the information technologies become important infrastructures, cyberattacks have become major concerns. These attacks not only compromise information system integrity but can also lead to substantial economic losses, reputation damage, and physical safety risks. Therefore, the information systems should be more robust against such cyberattacks.

To make the system more robust against attacks, it is important to comprehend how existing attacks emerge and spread by analyzing them in detail by reconstructing events of cyberattacks. Based on the knowledge obtained by the analysis of the existing attacks, we can make the system more robust against cyberattacks.

In this thesis, we first analyze ransomware-based cyberattacks using network forensics. We introduce a methodology to reconstruct the event chain of such attacks from packet capture data, which helps identify infected hosts and their routes of infection. We apply this method to the case of the CERBER ransomware. As a result, we found the event chain related to this ransomware; the user's search on bing.com led them to www.homeimprovement.com, a website compromised by cybercriminals. We also found that they used a pseudoDarkleech script to redirect visitors to a server that deployed the RIG Exploit Kit, resulting in the download of CERBER Ransomware.

While the initial parts of the thesis focused on traditional information systems such as servers, PCs, and smartphones, we also extended our analysis to systems based on machine learning models. With the widespread adoption of machine-learning and sensor technologies, these models have become new targets for cyberattacks. Therefore, protecting these models is crucial to ensuring robust defense mechanisms against such evolving cyber

threats.

Adversarial examples (AEs) are one of the largest vulnerabilities of the machine learning models. In this attacks, an adversary generates inputs that causes a machine learning system to generate incorrect outputs. Especially in the system using multiple sensors, some sensors may be vulnerable and can be used to generate AEs. Even if an attacker can attack a machine learning model by using only a small part of sensors, the vulnerabilities of sensors have a significant risk. However, the impact of hacking a small part of the sensor has not been discussed thus far.

Therefore, we also discuss the impact of a small part of sensors on the machine learning models and demonstrate that the attacker can change the output of the ML models using multiple sensors if the attacker can manipulate the values from a part of sensors in this thesis. We call this attack *sensor-based AEs*. We performed experiments using the human activity recognition model with three sensor devices attached to the chest, wrist, and ankle of a user, and demonstrate that attacks are possible by hacking one of the sensor devices.

Then, we also discuss the countermeasure against sensor-based AEs. One of the approach to protecting the system from the sensor-based AEs is to protect all sensor devices. However, it is difficult to protect all sensor devices, because the risk of the existence of the vulnerable sensor devices increases as the number of sensor devices increases. Therefore, we need a method to protect machine learning models even if a part of sensors are compromised by the attacker.

Therefore, in this thesis, we propose a new countermeasure focusing on the sensor-based AEs. This method detects sensor-based AEs and the sensors used by the attackers by checking the inconsistency of the output of the machine learning model obtained by changing the features used by the model. By detecting the sensors used by the attackers, we can check and replace them. Our method is based on the features of the sensor-based AEs that the attacker cannot avoid; the output of the machine learning model is altered when the values from the sensors used by the attacker are incorporated. We evaluated our method using a human activity recognition model with sensors attached to the user's chest, wrist, and ankle. We demonstrate that our method can accurately detect sensors used by

the attacker and achieves an average *Recall of Detection* of 0.92, and the average *Precision of Detection* is 0.72.

Acknowledgments

I am profoundly grateful for the guidance and support I received from many individuals during the course of my Ph.D. studies, each of whom has contributed immensely to my academic journey.

Foremost, I extend my deepest appreciation to my supervisor, Professor Masayuki Murata, for his invaluable guidance, insightful comments, and continuous encouragement throughout my research. His creative suggestions and patient mentorship have been pivotal in shaping my academic pursuits.

I would like to express my heartfelt thanks to the esteemed members of my thesis committee: Professor Takashi Watanabe, Professor Toru Hasegawa, and Professor Hirozumi Yamaguchi from the Graduate School of Information Science and Technology at Osaka University, along with Professor Hideyuki Shimonishi from the Cyber Media Center, Osaka University. Their comprehensive reviews and perceptive feedback have greatly enriched my work.

Special gratitude is owed to Associate Professor Yuichi Ohsita of Cyber Media Center, Osaka University for his dedicated mentorship and invaluable advice, which were crucial in the completion of my research.

I am also indebted to Associate Professor Shin'ichi Arakawa, Associate Professor Suyong Eum, Assistant Professor Daichi Kominami, and Assistant Professor Masaaki Yamauchi of Osaka University for their supportive comments and assistance. Their kindness and support have been indispensable to my academic progress.

My sincere thanks go to Mrs. Shihoko Kazama and Mrs. Kyoko Teramae for their

assistance and support throughout my doctoral program.

I wish to thank all members of the Advanced Network Architecture Research Laboratory at the Graduate School of Information Science and Technology, Osaka University, for their stimulating discussions and camaraderie.

A special acknowledgment to my in-laws, my mother, and my family, whose unwavering support and prayers have been a source of strength and inspiration.

Lastly, I reserve my deepest gratitude for my wife, Lucy Ademula, and my daughters, Siti Kumari Kanti Kaliani and Siti Kiaria Maznah. Their boundless support and encouragement throughout my Ph.D. studies have been a beacon of hope and strength, and I am eternally grateful for your promise of unwavering support in all my future endeavors.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As the information technologies become important infrastructures, cyberattacks have become major concerns [1]. These attacks not only compromise information system integrity but can also lead to substantial economic losses, reputation damage, and physical safety risks. A notable example of such a threat is the cyberattack on the UK Kingdom's National Health Service (NHS) in 2017 [2]. This attack exploits vulnerabilities in Windows operating systems and spreads through networks, encrypting data on infected computers, and demanding a ransom for decryption. Affecting over 200,000 computers in 150 countries, including numerous NHS systems, it has caused significant disruptions in healthcare services.

Therefore, the information systems should be more robust against such cyberattacks. Many methods to countermeasure against cyberattacks have been proposed. Some of them focus on the detection of cyberattacks [3–5]. In these methods, the features of the attacks are modeled as signatures and attacks are detected based on the signatures, or the normal behavior of the systems are modeled and anomalous traffic are detected. However, it is difficult to detect a new attack. To make the system more robust against attacks, it is important to comprehend how existing attacks emerge and spread by analyzing them in detail by reconstructing events of cyberattacks [6]. Based on the knowledge obtained by the analysis of the existing attacks, we can make the system more robust against cyberattacks.

In this thesis, we first discuss the analysis of the ransomware-based cyberattacks based on the network forensics. In this discussion, we propose an approach to reconstruct the event chain of a cyberattack from a packet capture data. By reconstructing the event chain, we can find the infected hosts and infected route.

The above discussion focuses on the traditional information systems based on servers, personal computers and smartphones. In recent years, the systems based on machine learning models have become widely used. In industrial process environments, the object detection based on machine learning models are used to make safety workspace [7]. In mining [8], the systems based on machine learning models ensures safer and more effective operations. In agriculture [9], the system based on machine learning models enables precise monitoring of soil and plant conditions, leading to improved yields. The systems based on the machine learning model are becoming used even in critical areas such as healthcare and safety [10, 11].

However, machine learning models have also become the target of attacks as machine learning and sensor technologies have become widely used. That is, we need to protect machine-learning models to create a robust system against cyberattacks.

Adversarial examples (AEs) are one of the largest vulnerabilities in machine-learning models. In these attacks, an adversary generates inputs that cause the machine learning system to generate incorrect outputs. Finlayson et al. showed that an adversary can exploit the vulnerability of machine learning (ML) models by creating AEs in critical domains, such as medicine [12]. These AEs contain subtle changes that are invisible to humans but can mislead the model's predictions. This poses a significant threat to systems based on ML models, particularly in sensitive fields.

In particular, in systems that use multiple sensors, certain sensors may be vulnerable and can be used to generate AEs. Manipulation of sensor data by compromising the software in the sensor device has been demonstrated [13]. Monjur et al. demonstrated that data manipulation is also possible by modifying the hardware if an attacker can physically access the sensor device [14]. Even if an attacker can attack a machine learning model using only a small number of sensors, the vulnerabilities of the sensors pose a significant risk. However,

the impact of hacking a small part of the sensor has not yet been discussed. Therefore, we also discuss the impact of a small part of sensors on the machine learning models and demonstrate that the attacker can change the output of the ML models using multiple sensors if the attacker can manipulate the values from a part of sensors in this thesis. We call this attack *sensor-based AEs*.

We also discuss countermeasures for sensor-based AEs. One approach to protecting the system from sensor-based AEs is to protect all the sensor devices. However, it is difficult to protect all sensor devices because the risk of vulnerable sensor devices increases as the number of sensor devices increases. Therefore, we need a method to protect machine-learning models, even if some sensors are compromised by the attacker.

Many countermeasures to mitigate the risks posed by AEs have been proposed [15–24]. However, existing methods can be avoided by modifying the AE generation process [25]. This is caused by the features used by the methods; the existing methods use features of the AEs, but an attacker with the knowledge of the countermeasure can change the features. That is, the detection method should use the features of the AEs that cannot be avoided by an attacker.

Therefore, in this thesis, we propose a new countermeasure focusing on the sensor-based AEs. This method detects sensor-based AEs and the sensors used by the attackers by checking the inconsistency of the output of the machine learning model obtained by changing the features used by the model. By detecting the sensors used by the attackers, we can check and replace them. Our method is based on the features of the sensor-based AEs that the attacker cannot avoid; the output of the machine learning model is altered when the values from the sensors used by the attacker are incorporated.

1.1 Malware and Ransomware

In the ever-evolving realm of cybersecurity, malware and ransomware have emerged as prominent threats, posing significant risks to individuals, organizations, and critical infrastructure. Understanding the nature of these threats and implementing effective mitigation

strategies is crucial for safeguarding sensitive data and ensuring the continued operation of critical systems.

Malware, an abbreviation for malicious software, encompasses a broad spectrum of software programs designed to harm a computer system or its users [26]. This diverse category includes viruses, worms, trojan horses, spyware, and ransomware, each with its unique modus operandi.

Ransomware, a particularly insidious form of malware, encrypts a victim's files, rendering them inaccessible [27]. This act essentially holds the victim's data hostage, with cybercriminals demanding a ransom payment in exchange for the decryption key. The financial repercussions of ransomware attacks can be severe, and the disruption to operations can be devastating [28].

The concept of ransomware can be traced back to 1989 with the creation of the PC Cyborg by Dr. Joseph Popp [29]. This early iteration concealed file folders and encrypted files on the `C:/ drive`, demanding a ransom for their release. Since then, ransomware has evolved considerably, incorporating sophisticated encryption techniques and becoming a major cybersecurity concern.

In recent years, several high-profile ransomware attacks have captured headlines, highlighting the severity of this threat. TeslaCrypt, Locky, and CERBER are just a few examples of notorious ransomware strains that have wreaked havoc on individuals and businesses [30]. A particularly notable instance occurred in 2016 when a Los Angeles hospital fell victim to a ransomware attack. Cybercriminals disabled the network and computers, demanding a ransom of $17,000 to restore the system containing sensitive patient information.

Ransomware typically spreads through network-based methods, often exploiting vulnerabilities in software or taking advantage of human error [31]. Common infection vectors include [28, 32]:

Downloading infected files: Malicious actors may distribute ransomware through

email attachments, file-sharing sites, or compromised websites. Clicking on a seemingly innocuous link or opening an infected file can trigger the ransomware installation.

Email phishing: Phishing emails often contain deceptive attachments or links that, when opened, unleash ransomware payloads onto unsuspecting users' systems.

Drive-by downloads: Visiting compromised websites can trigger the automatic download of ransomware, without any user interaction.

Exploit kits: These software packages exploit vulnerabilities in outdated software or operating systems to inject ransomware into vulnerable systems.

Several methods exist for detecting ransomware infections, each with its own strengths and limitations [3, 33, 34]:

- **Static feature-based methods:**These methods analyze the characteristics of a file or program to identify known patterns associated with ransomware. However, attackers can easily modify their malware to evade detection by these methods.
- **Dynamic-based methods:**These methods monitor the behavior of a running program to identify suspicious activities indicative of ransomware infection. While more effective than static methods, they can generate false positives and may not detect new malware samples.
- **Network behavior analysis:**This approach involves capturing and analyzing network traffic to identify patterns and anomalies that suggest the presence of ransomware or other malicious activity. It can provide valuable insights into the attack's propagation and potential targets

Network forensics plays a crucial role in investigating and responding to ransomware attacks. By analyzing network traffic, forensic investigators can reconstruct the events leading up to the attack, identify the initial infection point, and uncover the methods

used by the attackers. This information is invaluable for preventing future attacks and potentially recovering encrypted data.

Thus, in this thesis, we encompass analysis of ransomware, and creating and explaining a chain of cyberattack events.

1.2 Attacks on Machine Learning Models

The advancement of machine learning technologies, particularly in the field of deep neural networks, has been enhancing various aspects of human life, including security, efficiency, automation, and accuracy. Many applications of machine learning technologies have been used in many areas such as smart nations [35], agriculture [36], medicine [37], industry [38], and human activity recognition (HAR) [39].

However, deep neural networks are vulnerable to adversarial examples (AEs) [15], which are inputs designed to mislead AI models into producing incorrect outputs [40]. AEs can be generated by the gradient-based attacks [15, 41–44]. In these attacks, the attacker generates the input by adding perturbations calculated from the loss function's gradient or calculated by the optimization based on the adversarial training using GAN architectures.

Because systems based on machine learning models are being used even in critical areas such as medical AI systems, the impact of these attacks is also large. The possibility of AEs occurring in such critical areas has been demonstrated [12, 45, 46].

The risk of adversarial attacks escalates with the increasing number of sensor devices in a system if an attacker can generate AEs from some sensors. Therefore, in this thesis, we discuss the possibilities of such sensor-based AEs and demonstrate that the attacker can change the output of ML models using multiple sensors if the attacker can manipulate the values from a part of the sensors. In addition, we also discuss countermeasures against sensor-based AEs and propose a method to detect sensors used in sensor-based AEs.

1.3 Outline of Thesis

Analysis of Cerber Ransomware Behavior Based on Network Forensics [47]

Ransomware is becoming popular among cyber criminals. Ransomware are made more sophisticated and more effective as to avoid detection and analysis. In this thesis, we analyze ransomware-based cyberattacks using network forensics. We introduce a methodology to reconstruct the event chain of such attacks from packet capture data, which helps identify infected hosts and their routes of infection. We applied this method to the case of the CERBER ransomware. As a result, we found the event chain related to this ransomware; the user's search on bing.com led them to www.homeimprovement.com, a website compromised by cybercriminals. We also found that they used a pseudoDarkleech script to redirect visitors to a server that deployed the RIG Exploit Kit, resulting in the download of CERBER Ransomware.

Experiments on Adversarial Examples for Deep Learning Model Using Multimodal sensors [48]

Recently, artificial intelligence (AI) based on IoT sensors has been widely used, increasing the risk of attacks targeting AI. Adversarial examples are among the most serious types of attacks, in which an attacker designs inputs that can cause the machine learning system to generate incorrect outputs. Considering the architecture using multiple sensor devices, hacking even a few sensors can create a significant risk; an attacker can attack the machine learning model through the hacked sensors. Some studies have demonstrated the possibility of adversarial examples on deep neural network (DNN) models based on IoT sensors, but it was assumed that an attacker must access all features. The impact of hacking only a few sensors has not yet been discussed. Therefore, in this study, we discuss the possibility of attacks on DNN models by hacking only a small number of sensors. In this scenario, the attacker first hacks a few sensors in the system, obtains their values of the hacked sensors, and changes them to manipulate the system. However, an attacker cannot obtain and change the values of the other sensors. We performed experiments using the human

activity recognition model with three sensor devices attached to the chest, wrist, and ankle of a user and demonstrated that attacks are possible by hacking a small number of sensors.

Detection of Sensors Used for Adversarial Examples Against Machine Learning Models [49, 50]

Machine learning (ML) systems that use sensors obtain observations from the sensors and use them to recognize and interpret the current situation. Such systems are susceptible to sensor-based adversarial example attacks (AEs). If some sensors are vulnerable and can be compromised by an attacker, the attacker can change the output of the system by changing the values of the sensors. The detection of compromised sensors is important to defend the system against sensor-based AEs because we can check the sensors and replace them by detecting the sensors used by the attacker. In this thesis, we propose a method to detect the sensors used in sensor-based AEs by utilizing the features of an attack that cannot be avoided. In this method, we introduce a model called the feature-removable model (FRM), which allows us to select the features used as inputs into the model. Our method detects the sensors used in sensor-based AEs by determining inconsistencies between the outputs of the FRM obtained by changing the selected features. We evaluated our method using a human activity recognition model with sensors attached to the user's chest, wrist, and ankle. We demonstrate that our method can accurately detect the sensors used by the attacker and achieves an average *Recall of Detection* of 0.92, and the average *Precision of Detection* is 0.72.

In this thesis, we also discuss a strategy to make the system robust against sensor-based AEs proactively. A system with enough redundancy can work after removing the features from the sensors used in the AEs. That is, we need a metric to check if the system has enough redundancy. In this thesis, we define groups of sensors that might be compromised by the same attacker. Then, we also use the FRM to define a metric called *criticality* that indicates how important each group of sensors is for classification between two classes. Based on the criticality, we can make the system robust against sensor-based

AEs by interactively adding sensors so as to decrease the criticality of any groups of sensors for the classes that must be distinguished.

Chapter 2

Analysis of Cerber Ransomware Behavior Based on Network Forensics

2.1 Introduction

In 2016, a cyber attack on a Los Angeles hospital dramatically exposed the vulnerabilities in cybersecurity systems. This attack, involving ransomware, encrypted sensitive patient data and compelled the hospital to pay a $17,000 ransom to regain access [51]. This event is a stark example of the escalating threat posed by ransomware, which has become a preferred tool for cybercriminals engaged in financial extortion.

Ransomware, by design, encrypts the victim's data, denying them access until a ransom is paid for the decryption key [27, 51]. The growing reliance on digital data across various sectors has made such attacks increasingly common and destructive. This surge in ransomware incidents highlights the need for robust cybersecurity measures.

The ease of deployment and potential for high financial returns have driven the proliferation of ransomware. Cybercriminals can easily acquire ransomware kits from online

platforms, lowering entry barriers for initiating attacks. The success of these attacks is often guaranteed by the victims' willingness to pay ransoms, which inadvertently encourages the cycle of cyber extortion [28].

Beyond financial losses, ransomware attacks have broader implications, including operational downtime and reputational damage. The public nature of such breaches often leads to a loss of trust among customers and stakeholders. Equally important is the psychological impact on employees, manifesting as increased stress and decreased productivity, a consequence that is frequently overlooked [52, 53].

The year 2016 marked a significant increase in ransomware incidents, with rates soaring by 100-300%. The dramatic rise of 4000% in incidents from 2015 to 2016 underscores the growing sophistication and impact of these cyber threats [27, 54]. This period also saw the emergence of ransomware strains like TeslaCrypt, Locky, and CERBER, which have since dominated the cybercrime landscape.

The development of more complex and resilient ransomware variants to evade detection presents a formidable challenge to cybersecurity experts [28, 55, 56]. Traditional antivirus solutions, relying on static feature-based detection, are increasingly ineffective against these sophisticated techniques. Moreover, methods such as dynamic analysis in virtual machine environments are not foolproof, as contemporary malware can detect these environments and alter their behavior accordingly [4, 5, 33].

The case of Cerber ransomware exemplifies these challenges. Known for causing substantial damage and financial losses, Cerber typically infiltrates systems through network-based methods like file downloads, email phishing, and compromised websites [28]. A comprehensive understanding of these attacks, encompassing both detection and analysis, is crucial for developing effective countermeasures.

This thesis proposes a network forensics behavior-based approach to analyze Cerber ransomware attacks. This methodology goes beyond mere detection, focusing on reconstructing the sequence of events in a malware attack for both defensive and legal purposes [57]. By mapping the infection and dissemination process, this research aims to provide actionable insights for legal cases and recommendations for enhancing network security.

Employing this approach, the research identifies abnormal network traffic patterns to gain insights into the attackers' methodologies [58–60]. A thorough understanding of the malware attack lifecycle, from initial penetration to proliferation, is sought to contribute significantly to the fields of judicial and preventative cybersecurity.

Specifically, this study aims to reconstruct the events leading to a CERBER ransomware infection on a host computer named STIWIE PC. It involves tracing the initial infection vectors, such as the Trojan Godzilla and pseudoDarkleech script, and analyzing the ransomware payload used by cybercriminals.

The remaining sections of this chapter are structured for clarity and comprehensive coverage. Section 2.2 provides an in-depth exploration of ransomware and network forensics, with a particular focus on the Cerber variant. Section 2.3 outlines the methodology used in this research, detailing the hardware and software tools employed for the network forensics behavior-based analysis of Cerber ransomware. Section 2.4 discusses the results and key findings from this analysis. Finally, Section 2.5 concludes the chapter, summarizing the study's implications and suggesting directions for future research in this critical and evolving field.

2.2 Basic Theory

2.2.1 Ransomware

Ransomware has rapidly ascended as a predominant threat in the digital realm, presenting substantial risks to both individual users and organizations. This evolution from simple origins to a complex and widespread menace underscores the dynamic nature of cyber threats. Ransomware, notorious for encrypting vital files and extorting substantial ransoms for their release, has caused widespread disruption in both personal and professional spheres.

The inception of ransomware can be traced back to 1989 with the creation of PC Cyborg by Dr. Joseph Popp. This early form of ransomware targeted floppy disks, employing basic encryption methods and demanding a $189 ransom for file restoration [29]. Since this nascent stage, ransomware has undergone significant evolution, increasingly employing

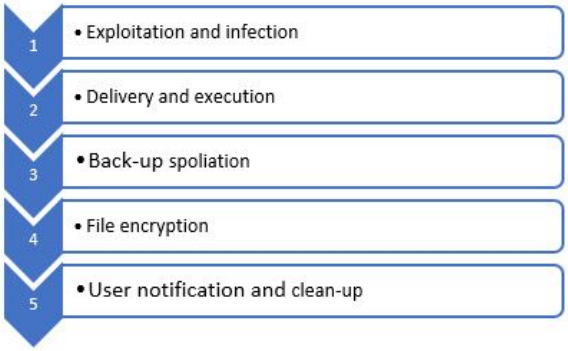


Figure 2.1: Five phases of ransomware

sophisticated techniques to outpace advancements in cybersecurity. Early forms combined virus and Trojan horse strategies, utilizing public key cryptography to secure encryption keys. These methods, known as "crypto virological attacks," represented a critical juncture in ransomware development, elevating both its sophistication and the level of threat posed [61].

As ransomware has evolved, it has become increasingly sophisticated, continuously adapting to counter cybersecurity measures. Contemporary ransomware attacks generally adhere to a structured five-phase cycle [28], as illustrated in Figure 2.1.

1. **Exploitation and Infection:** This phase marks the initiation of the attack, where the ransomware file is executed on the target computer. Phishing emails and exploit kits are common vectors, exploiting vulnerabilities in widely-used software like Adobe Flash and Internet Explorer. For instance, the notorious CryptoLocker attacks employed the Angler Exploit Kit for this purpose.

2. **Delivery and Execution:** Following successful exploitation, the ransomware executable is delivered and activated on the compromised system. It establishes persistence mechanisms to remain operational even after a system reboot. The executable is often encrypted, which hinders detection, and is strategically placed in directories like `%APPDATA%` or `%TEMP%`.

3. **Backup Spoliation:** In this stage, ransomware aims to eliminate backup files and

folders to thwart recovery efforts. Techniques like using the vssadmin tool to delete volume shadow copies are common, and some variants may aggressively delete backup folders, irrespective of their usage.

4. **File Encryption:** The malware conducts a secure key exchange with a command-and-control server to obtain encryption keys. Variants of ransomware exhibit different methodologies in file naming and encryption processes, which can be used to identify specific types of malware. The encryption duration is variable and depends on factors like network conditions, file size, and the number of infected devices.

5. **User Notification and Cleanup:** The final phase involves notifying the victim with ransom payment instructions, often under a time-sensitive deadline. These instructions may provide clues to the specific variant of ransomware used. Subsequently, the malware typically attempts to erase its traces from the system to complicate forensic analysis.

This comprehensive overview of ransomware's evolution, tactics, and attack methodologies highlights the critical need for vigilant cybersecurity practices. Understanding these phases and the methods employed by ransomware can aid in developing more effective defenses and response strategies against this ever-evolving cyber threat.

2.2.2 Cerber Ransomware

Cerber Ransomware stands as a sophisticated and formidable threat in the ever-evolving cybersecurity landscape. Operating under the Ransomware as a Service (RaaS) business model [62], this malicious software has wreaked havoc since its inception in Russia on March 4, 2016. Its primary mode of infection involves exploiting vulnerabilities in botnets, spam emails, and drive-by downloads [63]. Once Cerber gains access to a victim's system, it unleashes its potent AES encryption algorithm, effectively rendering the victim's data files inaccessible. To regain control over their precious data, victims are confronted with a chilling ultimatum: a ransom demand payable in digital currency, such as Bitcoin [64].

Cerber's nefarious tactics extend beyond mere encryptions. It employs a cunning strategy to identify the country of origin of each victim by scrutinizing IP geolocation data. If the victim's computer is located in one of the following countries: Armenia, Azerbaijan, Belarus, Georgia, Kyrgyzstan, Kazakhstan, Moldova, Russia, Turkmenistan, Tajikistan, Ukraine, Uzbekistan, Cerber will mercifully spare the system from encryption and terminate itself.

Upon successful execution in other countries, Cerber establishes a persistent presence within the victim's system by embedding itself deep within the '`%AppData%{2ED2A2FE-872C-D4A017ACE301404F1CBA}`' folder. To further entrench its hold, Cerber manipulates Windows boot settings, ensuring that it automatically launches into Safe Mode upon the next system reboot. This stealthy maneuver allows Cerber to operate undetected and execute malicious plans without interruption.

When the user logs into Windows, Cerber springs into action and seamlessly launches itself without alerting the user. It remains vigilant and continuously runs the screensaver when the system becomes idle. This tactic serves a dual purpose: to keep Cerber's presence hidden and to disrupt the normal workflow of the user, adding to their distress. In addition, Cerber bombards the user with bogus system alerts, further sowing confusion and panic. This relentless barrage of fake notifications aims to pressure the victim to pay the ransom, believing that their system is on the verge of collapse.

To ensure that the victim's desperation reaches its peak, Cerber leaves a trail of ominous notes in each encrypted folder: '`#DECRYPT MY FILES#.html`', '`#DECRYPT MY FILES#.txt`', and '`#DECRYPT MY FILES#.vbs`'. These cryptic messages serve as stark reminders of the victim's dire predicament and the looming threat of losing valuable data forever.

Cerber Ransomware represents a significant threat to organizations and individuals. Its sophisticated tactics and relentless pursuit of ransom payments make it a formidable adversary in the field of cybersecurity. Therefore, organizations must implement robust cybersecurity measures and educate their employees on the dangers of phishing emails and drive-by downloads to minimize the risk of Cerber infection. Additionally, maintaining regular backups of critical data provides a crucial lifeline in the event of a ransomware

attack.

2.2.3 Ransomware Detection Methods

Malware detection involves a range of techniques aimed at identifying malicious software in computer systems and networks. Broadly, these techniques fall into two main categories: static feature analysis and dynamic behavior analysis, as shown in Table 2.1. Each approach offers unique advantages in combating ransomware threats.

- **Static Feature Analysis**: This method entails examining the code and structure of a program or executable file. Static analysis detects patterns or characteristics indicative of malware, making it particularly effective for identifying known malware variants. It relies on signature-based detection, heuristic analysis, and file structure analysis. These techniques are instrumental in recognizing pre-established signatures or suspicious code structures, hence, valuable in catching well-documented malware strains. However, static analysis may fall short in detecting new, modified, or heavily obfuscated ransomware variants due to its reliance on pre-existing knowledge of malware characteristics [33].

- **Dynamic Behavior Analysis**: Unlike static analysis, dynamic behavior analysis monitors the behavior of a system or program during its execution. This method is crucial for identifying malware that attempts to conceal its presence or manipulate system processes. It involves observing real-time actions such as file encryption activities, changes to system files, or anomalous Windows API function calls. Dynamic analysis can detect novel and sophisticated ransomware strains, as it focuses on the behavioral patterns rather than pre-known signatures. This approach includes host behavior analysis, which monitors system behavior for anomalies, and network-based behavior analysis, which scrutinizes network traffic for signs of malicious activity like communication with remote servers or attempts to spread across the network [27, 56, 59].

While static and dynamic analyses are integral to ransomware detection, their scope is primarily limited to the identification of malware presence. Static analysis excels in quickly detecting known threats, and dynamic analysis offers insights into the behavior of malware, aiding in identifying newer or more complex ransomware variants. However, both methods predominantly focus on detection and do not delve into the deeper aspects of ransomware attacks, such as the precise timing, methodologies, and pathways of the infection.

This thesis introduces the use of network forensics, a more investigative approach that goes beyond mere detection. Network forensics allows for an in-depth exploration of the ransomware attack lifecycle, uncovering crucial details such as when, how, and through what means attackers infiltrate computer systems and networks. This comprehensive approach not only aids in understanding the specifics of ransomware attacks but also enhances the strategies for prevention and mitigation.

Table 2.1: Ransomware Detection Techniques

Technique	**Description**	**References**
Static	Signature-Based Detection: Identifies malware based on known signatures or patterns.	[65, 66]
Static	Heuristic Analysis: Examines code for suspicious characteristics indicative of malware.	[67]
Static	File Structure Analysis: Analyzes the structure of files to identify potential malware.	[68]
Dynamic	Monitored Windows API calls to identify ransomware behavior; focused on dynamic aspects of file manipulation.	[69]
Dynamic	Analyzed dynamic behavior patterns, including file encryption and API call sequences, to differentiate ransomware from legitimate processes.	[70, 71]
Dynamic	Focused on dynamic behavioral patterns of ransomware, particularly in how files are encrypted and manipulated.	[72]
Dynamic	Emphasized the importance of dynamic analysis in detecting ransomware through behavioral heuristics and pattern analysis.	[62]
Dynamic	Investigated dynamic ransomware detection through analysis of API function calls and user-level operations.	[73]

2.2.4 Network Forensics

Network forensics, a vital branch of digital forensics, employs rigorous scientific techniques to gather, analyze, and preserve digital evidence from network traffic. It is essential in identifying, linking, and examining evidence from a variety of sources, including electronic devices and digital networks [74–76].

A landmark study in the application of network forensics to ransomware analysis was conducted in "This Thesis" by Kurniawan (2018) [47]. This research stands out as the first to apply a network forensic methodology specifically to the study of ransomware. Focusing on the Cerber Ransomware, the study emphasizes a behavioral-based network forensic approach. It aims to reconstruct attack timelines, identify infected hosts, and analyze the chain of infection, highlighting the increasing sophistication of ransomware.

Network forensics are invaluable in capturing network traffic to and from specific hosts, shedding light on the channels, methods, and spread of malicious code [76, 77]. This analysis enables investigators to reconstruct event sequences, identify compromised systems, and trace the origin of attacks.

Despite its utility, network forensics faces unique challenges. The dynamic and ephemeral nature of network-based data makes evidence gathering more complex compared to traditional digital forensics, which deals with static storage devices [59, 78]. This necessitates sophisticated collection techniques.

Furthermore, the interpretation of network forensic evidence is challenging due to the vast amount of data involved. Understanding this data requires a deep knowledge of network protocols, communication patterns, and potential attack vectors.

Investigators in network forensics often work with live systems, like routers and servers, which complicates evidence gathering [59]. These systems must remain online to avoid disrupting critical operations, requiring real-time evidence collection while maintaining network functionality.

Despite these challenges, network forensics remains an indispensable tool in cybersecurity. By analyzing network traffic, investigators can uncover traces of cybercrime, gaining

Table 2.2: Network Forensics and Ransomware Attacks

Authors	Brief Description
This Thesis (Kurniawan, 2018) [47]	The first study to employ network forensic methods in ransomware analysis, focusing on the Cerber Ransomware. Emphasizes a behavioral-based approach to reconstruct attack timelines, identify infected hosts, and analyze infection chains. Highlights the increasing sophistication of ransomware.
Paul Joseph & Norman, 2020 [79]	Reviews the role of memory forensics in analyzing ransomware, specifically WannaCry. Focuses on the importance of memory forensics, its tools, and practical application on affected computers. Discusses in-depth analysis techniques like DLL tracing and reverse engineering.
Umar et al., 2021 [80]	examine Conti ransomware, focusing on its network behavior and impact. Uses a three-stage process: simulation, network forensics via live methods, and malware analysis. The study results in detailed forensic data useful for identifying ransomware traffic and coping with zero-day threats.
Surya Kusuma et al., 2021 [81]	Focuses on reconstructing Ryuk ransomware attacks and analyzing infection sources using the TAARA method. Utilizes tools like Wireshark for forensic data collection. Achieves detailed insights into the attack source, timeline, and infection process.

insights into attack methods, compromised systems, and potential perpetrators. This information is crucial for preventing future attacks, bringing perpetrators to justice, and protecting sensitive data.

Comprehensive Overview of Ransomware Research in Network Forensics

We present a comprehensive overview of distinct research papers, as shown in Table 2.2. These studies represent various methodologies, addressing different ransomware types and offering insights into their operational mechanisms and network impact.

.

Figure 2.2: OSCAR Methodology: A Comprehensive Framework for Network Forensics Investigations

2.3 Methods

Preparation Stage

The preparation stage involved the establishment of the hardware and software resources required for this study. The hardware employed consists of a Notebook Processor: Intel® Core™ i7-6500U CPU @ 2.30GHz, 8GB RAM, 250GB SSD, and an Intel 530 Graphics Card. The software utilized includes Wireshark Version 2.2.5 and a malware traffic dataset obtained from `http://www.malware-traffic-analysis.net/`.

2.3.1 OSCAR Methodology: A Comprehensive Framework for Network Forensics Investigations

The OSCAR methodology [60], is shown in Figure 2.2. OSCAR an acronym for Obtain, Strategize, Collect, Analyze, and Report, serves as a structured framework for conducting network forensics investigations. This systematic approach ensures that investigations are conducted in a rigorous and defensible manner, maximizing the likelihood of successful outcomes.

1. **Obtain Information**

   The initial phase of the OSCAR methodology involves gathering comprehensive information about the incident and the environment in which it occurred. This crucial step entails understanding:

- **Incident Details:** Accurately understanding the nature of the incident, including its date, time, method of discovery, individuals involved, affected systems and data, actions taken to date, and any relevant summaries of internal discussions.
- **Environmental Context:** Gaining a thorough understanding of the organization's network topology, available sources of network evidence, organizational structure, incident response procedures, communication systems, and available resources, including personnel, equipment, funding, and time constraints.

2. **Strategize**

   Based on the gathered information, the next step involves strategizing the investigation approach. This includes:

   - **Evidence Prioritization:** Carefully evaluating the potential value, acquisition effort, and volatility of each evidence source to prioritize those most likely to yield relevant and reliable data.
   - **Acquisition Planning:** Developing a detailed plan for acquiring evidence from each prioritized source, considering factors such as evidence volatility, available resources, and chain of custody requirements.

3. **Collect Evidence**

   With a well-defined strategy in place, the evidence-collection phase commences, adhering to strict protocols to ensure data integrity and chain of custody maintenance. This involves:

   - **Thorough Documentation:** Maintaining a meticulous log of all systems accessed, actions are taken, and evidence-handling procedures to ensure transparency and audibility.
   - **Evidence Capture:** Employing appropriate techniques to capture evidence from each source, such as copying logs, imaging hard drives, or capturing network

packets.

- **Secure Storage:** Storing evidence in a secure and controlled environment, maintaining the chain of custody to preserve its admissibility in legal proceedings.

4. **Analyze**

   The collected evidence undergoes rigorous analysis to identify patterns, trends, and anomalies that may shed light on the incident. This entails:

   - **Evidence Correlation:** Correlating data from multiple sources to establish connections and gain a holistic understanding of the incident timeline.
   - **Timeline Construction:** Creating a detailed timeline of events, including timestamps, actions taken, and individuals involved, to reconstruct the sequence of events.
   - **Event Identification:** Identifying and prioritizing events of interest that are most relevant to the investigation and may provide clues about the root cause and perpetrator.
   - **Evidence Corroboration:** Corroborating evidence from different sources to enhance its reliability and credibility.
   - **Additional Evidence Recovery:** As new insights emerge, identifying and recovering additional evidence to supplement the existing dataset.

5. **Report**

   The culmination of the OSCAR methodology is the comprehensive investigation report, which communicates the findings and conclusions to stakeholders. The report should:

   - **Target a Nontechnical Audience:** Utilize a clear and concise language that is understandable to individuals without a technical background.

- **Maintain Defensibility:** Base the report on verifiable evidence and provide supporting documentation to uphold its credibility in legal or administrative settings.
- **Adhere to Factual Reporting:** Avoid speculation or personal opinions, ensuring that the report presents an objective and unbiased account of the investigation.

By diligently following the OSCAR methodology, investigators can conduct network forensics investigations with rigor, efficiency, and defensibility, maximizing the likelihood of uncovering the truth and bringing perpetrators to justice.

2.4 Results

This study meticulously examines the intricate processes underlying ransomware infections and their dissemination across network systems. A critical preventive strategy for organizations is the deployment of packet capture tools. These tools assiduously record all network traffic, enabling the identification of malicious activities, whether internal or external, regardless of their origin. The captured data serve as indispensable digital evidence during forensic investigations of legal transgressions.

2.4.1 Analysis

Timestamps play a crucial role in digital forensics as they provide information about the specific time of events [59]. Our detection and forensic analysis, were conducted using the Wireshark Network with an HTTP.request filter, aimed to ascertain the initial infection time of the host computer. As illustrated in Figure 2.3, the infection first occurred on January 27, 2017, at 22:53:54 UTC (January 28, 2017, 05:53:54 SE Asia Standard Time).

Identifying the date and time of infection led to the next phase of detecting and analyzing the IP address, MAC Address, and hostname of the infected computer. For this purpose, we used a NetBIOS Name Service `(NBNS)` filter. NetBIOS, a critical application for network

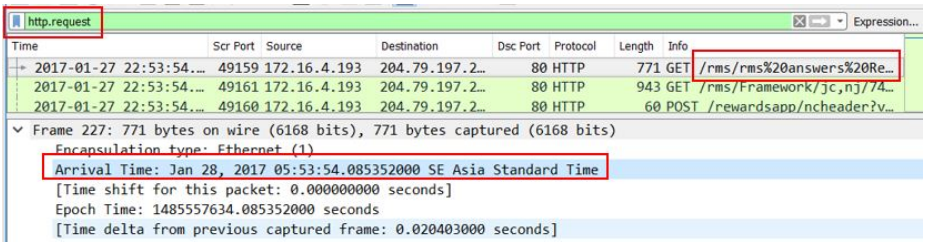


Figure 2.3: Timestamp of the initial infection

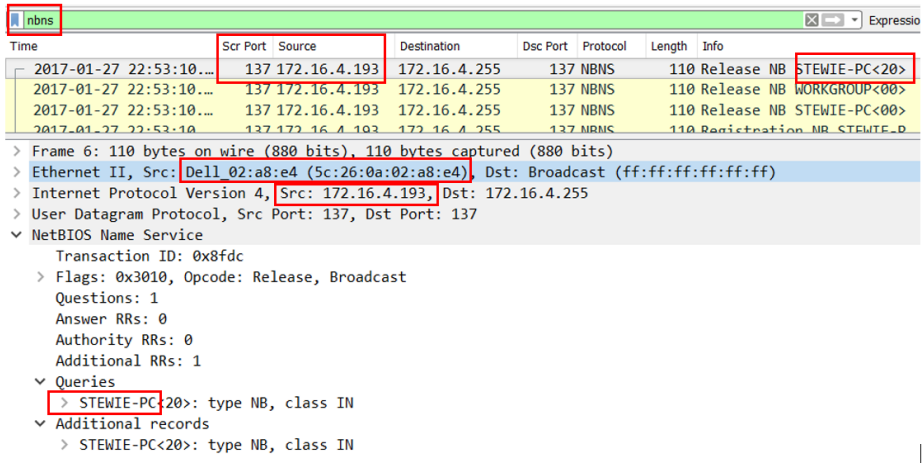
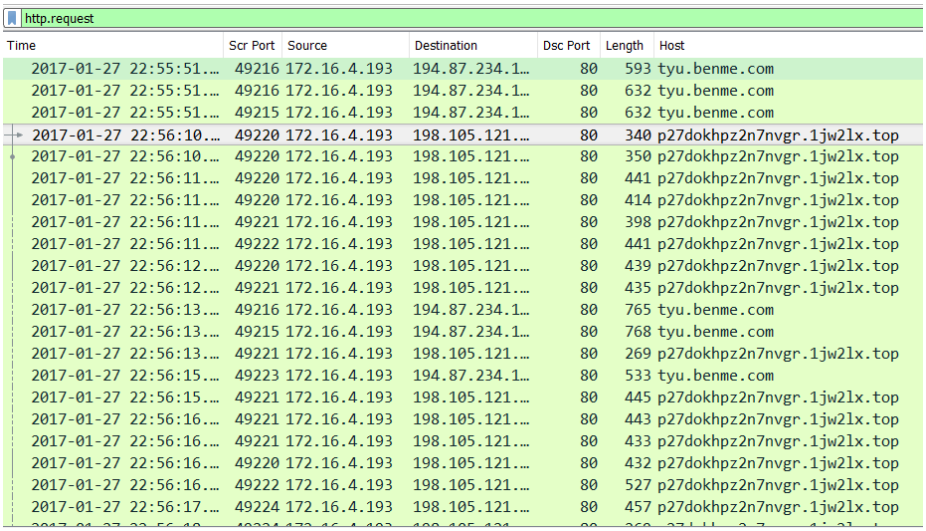


Figure 2.4: NBNS Traffic Analysis

communication, particularly in LAN environments, facilitated this analysis. Figure 2.4 displays the infected victim's computer details, identifying the IP as `172.16.4.193`, with the MAC Address `5c:26:0A:02:a8:e4`, a `Dell` network card, and the hostname `Stewie PC`.

Once we had identified the IP, MAC Address, and hostname, our next objective was to determine the specific malware infecting `Stewie PC`. In-depth packet analysis, as shown in Figure 2.5, revealed traffic to the `domain.top`, a common domain used by malware authors for criminal activities. A list of commonly used domains includes various domains. onion links that are known to be associated with malicious activities. List of Domains: `lclebb6kvohlkcml.onion`, `bmacyzmea723xyaz.onion`, and `nejdtkok7oz5kjoc.onion`.

Our analysis led to the discovery of a specific domain that is utilized by cybercriminals. Using Google's search engine with the keyword `p27dokhpz2n7nvgr.1jw2lx.top,` we found



http.request

Time	Scr Port	Source	Destination	Dsc Port	Length	Host
2017-01-27 22:55:51.…	49216	172.16.4.193	194.87.234.1…	80	593	tyu.benme.com
2017-01-27 22:55:51.…	49216	172.16.4.193	194.87.234.1…	80	632	tyu.benme.com
2017-01-27 22:55:51.…	49215	172.16.4.193	194.87.234.1…	80	632	tyu.benme.com
2017-01-27 22:56:10.…	49220	172.16.4.193	198.105.121.…	80	340	p27dokhpz2n7nvgr.1jw2lx.top
2017-01-27 22:56:10.…	49220	172.16.4.193	198.105.121.…	80	350	p27dokhpz2n7nvgr.1jw2lx.top
2017-01-27 22:56:11.…	49220	172.16.4.193	198.105.121.…	80	441	p27dokhpz2n7nvgr.1jw2lx.top
2017-01-27 22:56:11.…	49220	172.16.4.193	198.105.121.…	80	414	p27dokhpz2n7nvgr.1jw2lx.top
2017-01-27 22:56:11.…	49221	172.16.4.193	198.105.121.…	80	398	p27dokhpz2n7nvgr.1jw2lx.top
2017-01-27 22:56:11.…	49222	172.16.4.193	198.105.121.…	80	441	p27dokhpz2n7nvgr.1jw2lx.top
2017-01-27 22:56:12.…	49220	172.16.4.193	198.105.121.…	80	439	p27dokhpz2n7nvgr.1jw2lx.top
2017-01-27 22:56:12.…	49221	172.16.4.193	198.105.121.…	80	435	p27dokhpz2n7nvgr.1jw2lx.top
2017-01-27 22:56:13.…	49216	172.16.4.193	194.87.234.1…	80	765	tyu.benme.com
2017-01-27 22:56:13.…	49215	172.16.4.193	194.87.234.1…	80	768	tyu.benme.com
2017-01-27 22:56:13.…	49221	172.16.4.193	198.105.121.…	80	269	p27dokhpz2n7nvgr.1jw2lx.top
2017-01-27 22:56:15.…	49223	172.16.4.193	194.87.234.1…	80	533	tyu.benme.com
2017-01-27 22:56:15.…	49221	172.16.4.193	198.105.121.…	80	445	p27dokhpz2n7nvgr.1jw2lx.top
2017-01-27 22:56:16.…	49221	172.16.4.193	198.105.121.…	80	443	p27dokhpz2n7nvgr.1jw2lx.top
2017-01-27 22:56:16.…	49221	172.16.4.193	198.105.121.…	80	433	p27dokhpz2n7nvgr.1jw2lx.top
2017-01-27 22:56:16.…	49220	172.16.4.193	198.105.121.…	80	432	p27dokhpz2n7nvgr.1jw2lx.top
2017-01-27 22:56:16.…	49222	172.16.4.193	198.105.121.…	80	527	p27dokhpz2n7nvgr.1jw2lx.top
2017-01-27 22:56:17.…	49224	172.16.4.193	198.105.121.…	80	457	p27dokhpz2n7nvgr.1jw2lx.top

Figure 2.5: Information Gathering

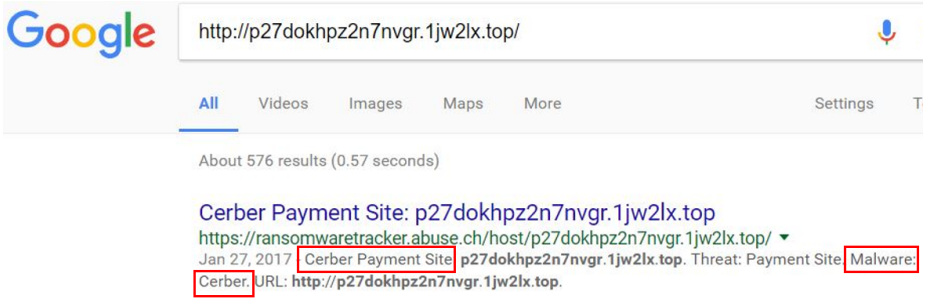


Figure 2.6: Google Search Results for CERBER Ransomware

that the malware infecting `Stewie PC` was CERBER Ransomware, as depicted in Figure 2.6.

In Figure 2.7, we present a detailed analysis of the PCAP data that has been meticulously uploaded to `https://www.virustotal.com`, a renowned online platform for virus and malware detection. The figure prominently features an alert from Suricata, which is an advanced network threat detection engine. This alert is critical, as it reveals the discovery of a notorious ransomware, identified as Cerber, actively utilizing the RIG Exploit Kit (EK). RIG EK is a sophisticated and widely used toolkit in the cybercriminal world to exploit vulnerabilities in systems to distribute malware. Our analysis in Figure 2.7 not only shows the effectiveness of Suricata in identifying such threats but also highlights the

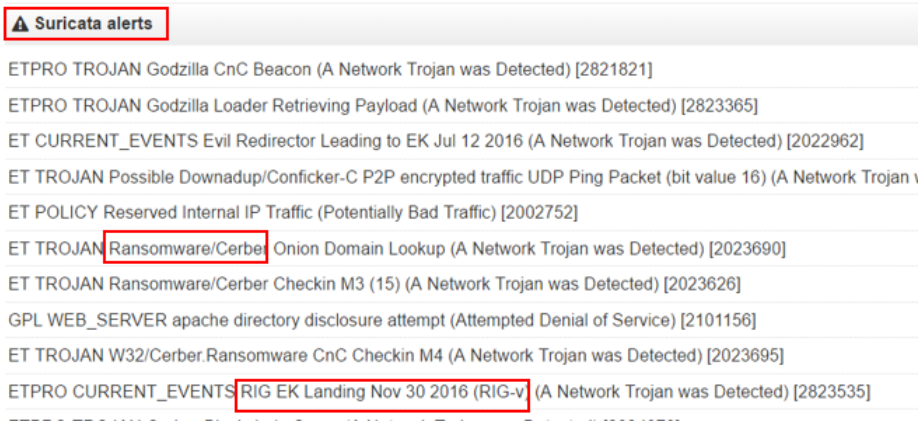


Figure 2.7: The PCAP result uploaded to `https://www.virustotal.com`, highlighting Suricata's detection of the Cerber cybercriminal using RIG EK.

Figure 2.8: Snort Result for RIG Exploit Kit Detection

intricate patterns and methods employed by RIG EK, providing valuable insights into the evolving landscape of cyber threats and the importance of proactive detection and defense mechanisms in cybersecurity.

Another important discovery was made when the `Pcap` file was analyzed using Snort, as shown in Figure 2.8. We detected an RIG exploit kit's landing page. Exploit Kits (EKs) are server-based frameworks that exploit software vulnerabilities, often associated with web browsers, to infect victims' machines without their awareness. In particular, RIG EK serves as a gateway for the delivery and distribution of malware payloads.

In Figure 2.9, we demonstrate the result of filtering `http.request` and `ip.addr194.87.234.129`, which highlights the IP address associated with Rig EK. Ransomware typically

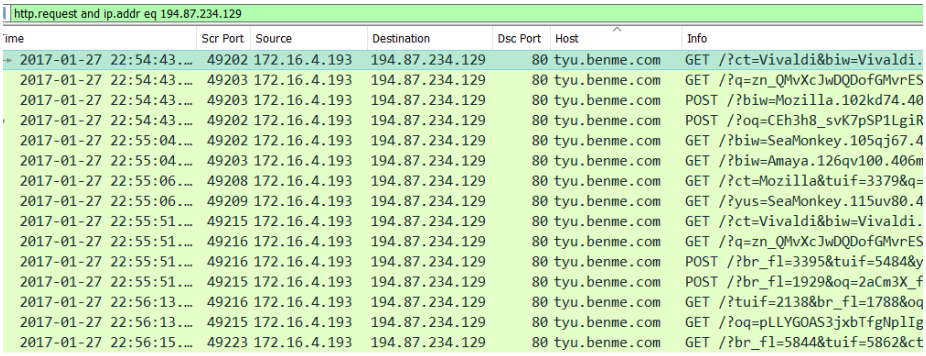


Figure 2.9: HTTP Requests to the Rig Exploit Kit Internet Protocol Address

Figure 2.10: Follow HTTP Stream to Find Referrer

spreads through two methods: malicious spam (mail spam) and Exploit Kits. Malicious spams directly target victims, prompting them to click on infected links or attachments. By contrast, Exploit Kits operate covertly, automating the exploitation of security vulnerabilities without requiring active victim participation.

The final phase involves filtering HTTP requests for all IP addresses associated with Rig EK in Wireshark. This phase aimed to detect and analyze RIG EK and the domain website that mediates the spread of infection. The results, as shown in figure 2.10, revealed that the host computer was accessing `www.homeimprovement.com`. Further analysis indicated that the victim accessed this site while searching for `"remodeling your kitchen cabinets"` on `https://www.bing.com/`.

In conclusion, our analysis confirmed that `www.homeimprovement.com` was a compromised website used in spreading RIG EK. This malware distribution method is sophisticated and involves various stages and components. The final figure, Figure 2.11, illustrates the

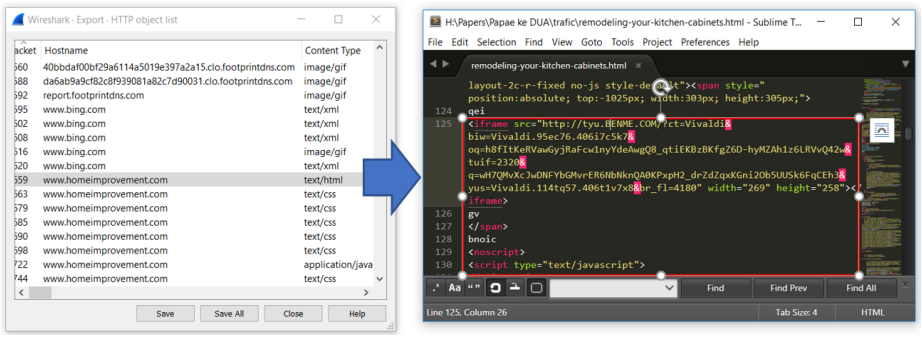


Figure 2.11: Export Object List and PseudoDarkleech Script

Figure 2.12: PseudoDarkleech Campaign

pseudoDarkleech script, a common campaign used by Cerber authors, designed to stealthily redirect traffic to the Exploit Kit server.

The pseudoDarkleech campaign, as detailed in Figure 2.12, followed a specific chain of events. Initially, the victim visits a compromised website, which then makes an HTTP request to the Exploit Kit Landing Page. The EK then assesses the computer for vulnerabilities, primarily in browser-based applications and Adobe Flash Player; if successful, it delivers the ransomware payload for encryption and file access.

Overall, this study underscores the complexity of ransomware attacks and the importance of thorough digital forensic practices in understanding and combating cyber threats.

2.5 Conclusion

This research makes significant contributions to the field of cybersecurity, particularly in the context of network forensics and the analysis of sophisticated cyber threats like Cerber Ransomware. Through the application of network forensic-behavior-based methods, this study has successfully identified and reconstructed the event chain of the Cerber Ransomware attack, as illustrated in Figure 2.13.

Figure 2.13: The chain of events of a host PC was infected with cerber ransomware.

The sequence of events begins with the STIWIE PC host computer, where an innocent search on Bing.com leads the user to the compromised website, www.homeimprovement.com. This website, tainted by a pseudoDarkleech script, plays a pivotal role in the cybercriminals' campaign, redirecting victims to a server that deploys the RIG Exploit Kit (EK). This exploit kit is crucial for the downloading of the Cerber Ransomware malware payload, highlighting a sophisticated method of cyber attack.

The findings emphasize the critical need for heightened awareness and proactive measures against cyber threats. Users of host PCs must regularly update their system's security, and network defenses, and patch vulnerabilities to mitigate such risks.

Further, in-depth network forensic analyses are essential for compromised websites and servers hosting Exploit Kits, particularly because these kits feature encrypted binary codes that pose challenges in detection and analysis.

However, it is crucial to note that network forensics alone may not suffice to address adversarial examples in sensor and machine-learning integrations. Adversarial attacks present unique challenges that require specialized countermeasures. Therefore, future research will

focus on developing alternative strategies to counter adversarial attacks in sensor ML-integrated systems. This includes exploring proactive measures for sensor robustness and ML model resilience to mitigate the impact of sophisticated attacks. By understanding the dynamics between sensors and ML models, we can devise more effective defenses that go beyond traditional network forensics, thereby providing a comprehensive shield against these emerging threats. Building a 'Human Firewall' remains essential, but it should be complemented with advanced technical strategies to strengthen the overall security infrastructure.

Chapter 3

Experiments on Adversarial Examples for Deep Learning Model Using Multimodal Sensors

3.1 Introduction

Artificial intelligence (AI) based on deep neural networks (DNNs) has significantly impacted human lives by making them more secure, efficient, automated, and accurate. Currently, AI is widely used in many areas, such as smart nations [35], agriculture [36], medicine [37], industry [38], and human activity recognition (HAR) [39]. Many Internet of Things (IoT) sensor devices are used to achieve accurate recognition of the real world. Observations from these devices are collected via the Internet or a network managed by the service provider. The machine learning model then recognizes the current situation using the collected observations as its input. For example, an autonomous vehicle recognizes the surroundings using multimodal sensors, such as cameras, radar, LiDAR, global navigation satellite system (GNSS), gyroscopes, and magnetometers [82]. Many sensors have been used in HAR. Ichino et al. used accelerometers, gyroscopes, magnetometers, and electrocardiogram sensors to recognize human activities [83]. Debauche et al. also proposed a

model to recognize human activities based on accelerometer and gyroscope signals [84].

Although we celebrate advances in AI and sensors, state-of-the-art DNNs are vulnerable to adversarial examples [15]. Adversarial examples are inputs designed by an adversary that cause a machine learning system to generate incorrect outputs [40]. By creating an incorrect output, an attacker can degrade the service that is based on AI. If the service is related to users' health, the degradation of the service may have a significant impact on the users' health. By attacking such services, an attacker may consider users as hostages. Some studies have demonstrated that an attacker can fool machine learning models for HAR, which is closely related to healthcare [46, 85, 86].

Considering an architecture using multiple sensor devices, hacking a small number of sensors creates a significant risk. As the number of sensor devices in a system increases, the risk of attack by hacking some of these sensors increases. Some sensors may be located near people. As a result, attackers may easily access them, find vulnerabilities, or replace them. In fact, an attack on wearable sensors has already been demonstrated [13].

The attacker may attack the machine learning model using the hacked sensor. Even if an attacker can hack only a small part of the sensor, the sensor may have a large impact on the machine learning model. However, the impact of hacking a small part of the sensor has not been discussed thus far. Some papers demonstrated that adversarial examples on the DNN model based on IoT sensors are possible, but with the assumption that an attacker can access all features of the model.

Therefore, we discuss the possibility of attacks on DNN models by hacking a small number of sensors through experiments. In this experiment, we assume that the attacker first hacks the sensor device. The attacker can obtain the values of the hacked sensors and change them but cannot obtain and change the values of the other sensors. In this study, we demonstrate that an attacker can manipulate a DNN model, even in this case.

To demonstrate the attacks, we introduce a generator that generates adversarial examples when a small number of sensor devices are hacked. The generator uses the values from the hacked sensors as inputs and generates perturbations so that the features, including the perturbations, are classified into the target class by the target model. In our experiments,

we use an open dataset for HAR based on three sensor devices attached to the chest, wrist, and ankle of the subjects and demonstrate that the attacker can change the output of the target model by hacking only one of those devices.

In summary, our main contributions are as follows:

1. We formulate the adversarial example by monitoring and changing the values of a part of the sensors.

2. We demonstrate that adversarial examples are possible even if the attacker can monitor and change only a part of the sensors.

The rest of this article is organized as follows. We discuss the work related to our research in Section 3.2. In Section 3.3, we describe the attack definition and how to generate the attacks. Section 3.4 presents adversarial attack examples for deep learning models using multimodal sensors. The results are discussed in Section 3.5. Finally, we conclude the chapter in Section 3.6.

3.2 Related Work

Szegedy et al. coined the phrase "adversarial example," and since then the number of publications related to adversarial examples has increased exponentially.

Several methods for generating adversarial examples have been proposed, as shown in Table 3.1, such as the fast gradient sign method (FGSM) [15], basic iterative method (BIM) [41], saliency map method [42], FGSM [15], and Carlini–Wagner method (C&W) [43] and AdvGAN [44]. FGSM and BIM are examples of white-box attacks that access an entire target model. Gradient-based attack methods, such as FGSM, determine the maximum constrained max-norm perturbation of $x' = x + \epsilon \cdot \text{sign}(\nabla_x L(x, y))$ by computing the gradient of the input's loss function $\nabla_x L(x, y)$ and multiplying a small chosen constant ϵ by the gradient's sign vector. Carlini and Wagner attacks [43] and other optimization-based methods optimize adversarial perturbations, subject to several constraints. These methods focus on the L^0, L^2, and L^∞ distance metrics and generate perturbations by minimizing the

loss function under the constraint that the distance metrics of the perturbations are less than a predefined threshold. Although optimization-based methods generate adversarial perturbations that fool the target model without violating the constraints, they take a long time because the optimization problem must be solved to generate each perturbation.

Another approach for generating adversarial examples is to train the generator. Xiao et al. [44] proposed adversarial examples using GAN architecture to efficiently generate more realistic adversarial examples. This was followed by [87–89]. They proposed training a feed-forward network that generates perturbations to create diverse adversarial examples and a discriminator network to ensure that the generated examples are realistic. Once the generator is trained, adversarial perturbations can be efficiently generated.

In this study, we used a method based on the generator. However, we assume that an attacker can obtain only the values of the hacked sensors, whereas existing studies assume that all features can be used as the input of the generator, as shown in Table 3.1.

Potential adversarial examples have also been discussed in many critical applications. Table 3.2 shows the papers demonstrating adversarial examples. Finlayson et al. demonstrated the danger of adversarial attacks in the medical domain [12]. By taking input from the vision sensor and adding adversarial noise to a dermatoscopy image, they successfully changed the patient's diagnosis from benign to malignant or vice versa. Han et al. demonstrated an attack on a deep learning model based on a raw signal electrocardiogram (ECG) [45]. Benegui et al. successfully attacked a DNN model for user identification based on motion sensors and converted a discrete three-axis raw signal sensor into a grayscale image representation [85]. Sah et al. demonstrated attacks on a machine learning model for HAR based on multiple wearable sensors [86]. They generated adversarial examples at the raw signal level and discussed their transferability. In this study, we use the same dataset as Sah et al. to demonstrate the possibility of an attack, but the attack scenario is different; we assume that the attacker can only access the hacked sensors, whereas Sah et al. assumed that the attacker could access all raw signals directly.

Researchers	Name of the Proposed Method	Advantages	Disadvantages
Goodfellow et al. [15]	FGSM	One of the first attack methods in the domain of adversarial examples	All features are required; low attack success rate
Kurakin et al. [41]	BIM	Higher attack success ratio than FGSM	All features are required; many iterations are required
Papernot et al. [42]	Saliency map	Attack with a small perturbation	All features are required; computationally expensive
Carlini and Wagner [43]	C&W	Higher attack success ratio than FGSM, BIM, and saliency map	All features are required; computationally expensive
Xiao et al. [44] Jandial et al. [87] Liu et al. [88] Kim et al. [89]	GAN	Higher attack success ratio than FGSM and C&W in the case that adversarial training is used to protect the target model; the perturbations can be generated immediately by using the pretrained generator	All features are required; training is required before generating the attack
This study	Trained generator	The attack can be generated even when the attacker can monitor only a part of the features; the perturbations can be generated immediately by using the pretrained generator	Training is required before generating the attack

Table 3.1: Advantages and disadvantages of previous methods

Researchers	Methods are Used to Generate Attacks	Target Features	Features Required to Be Monitored and Changed	Detail
Finlayson et al. [12]	BIM	Images	All	Demonstration of adversarial examples against medical AI systems
Han et al. [45]	FGSM and BIM	Raw sensor values	All	Demonstration of adversarial examples on raw EEG signals; the generated signals cannot be distinguished from original ECG signals and can fool the target DNN model
Benegui et a. [85]	FGSM and saliency map	Images	All	The first study attempts to quantify the effect of adversarial assaults on machine learning models used for motion sensor-based user identification
Sah et al. [46]	FGSM and BIM	Raw sensor values	All	Demonstration of the transferability of adversarial examples on machine learning models based on wearable sensors
This study	Trained generator	Raw sensor values	Part	Demonstrates adversarial examples for the case that only a part of the features is monitored by the attacker

Table 3.2: Summary of Research in Adversarial Machine Learning

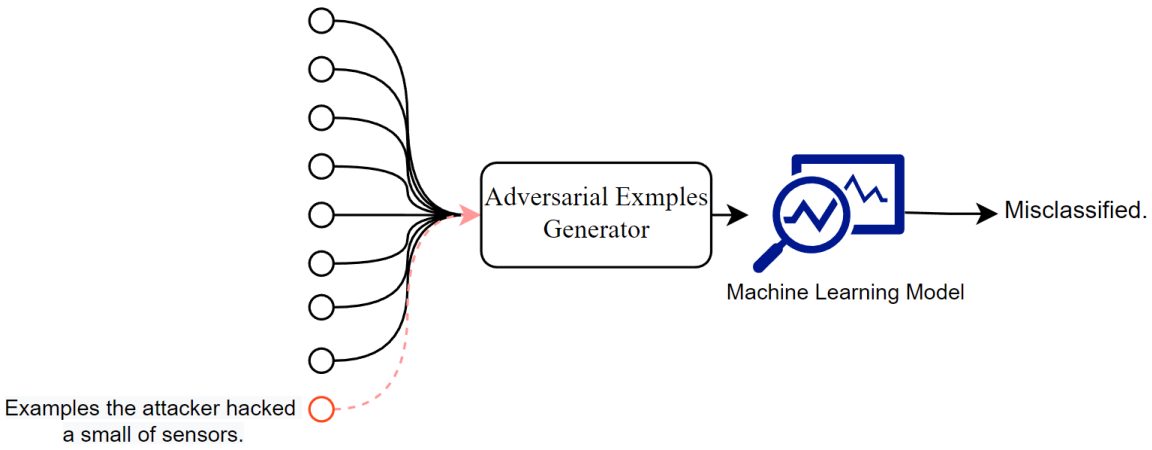


Figure 3.1: Overview of the attacks

3.3 Definition of Adversarial Examples by Hacking a Small Number of Sensors

3.3.1 Definition of Attack

We focus on a system that gathers values from multiple sensors and performs classification tasks based on a machine learning model. An attacker against this system hacks some sensors; the attacker may hack sensors with the same vulnerabilities but cannot hack other sensors. An attacker can obtain the values of the hacked sensors and change them. The objective of the attack is to cause misclassification by changing the values of the hacked sensors, as shown in Figure 3.1

Hereafter, we define $f(x_{0:t})$ as a function of the target model; $x_{0:t} = (x_0, x_1, \ldots, x_t)$ is the input of the target model constructed from the sensor values obtained from time 0 to time t'; and x_t is the vector corresponding to the sensor values at time t. $f(x_{0:t})$ indicates the classification result at time t; we denote the jth element of the output of the model by $f_j(x_{0:t})$, and $f_j(x_{0:t})$ indicates the probability that the situation at time t is classified into the ith class.

The attacker can monitor and change the features from the hacked sensors. We define the vector $B = (b_1, b_2, \ldots, b_m)$ indicating the features from the hacked sensors; $b_i = 1$ if the

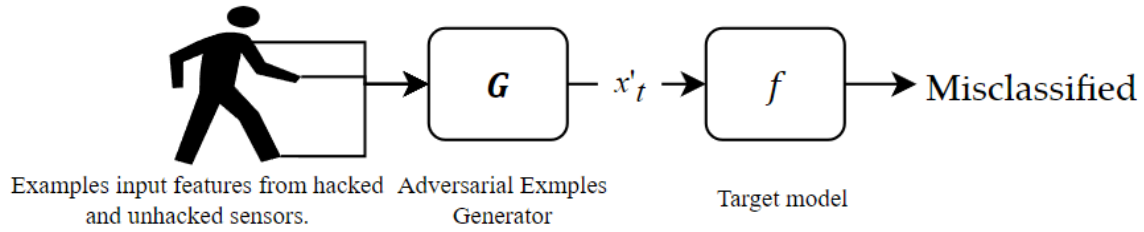
ith feature is from the hacked sensor. Using B, the features monitored by the attacker at time t are $\dot{x}_t = B \circ x_t$, where $\circ$ indicates the element-wise product.

The attacker generates the perturbations so that the classification results become the target class. That is, the objective of the attacker is $\arg\max_i f_i(x'_{0:t}) = C$, where $x'_{0:t} = (x'_0, x'_1, \ldots, x'_t)$ is the input feature after adding the perturbation at time t and C is the target class.

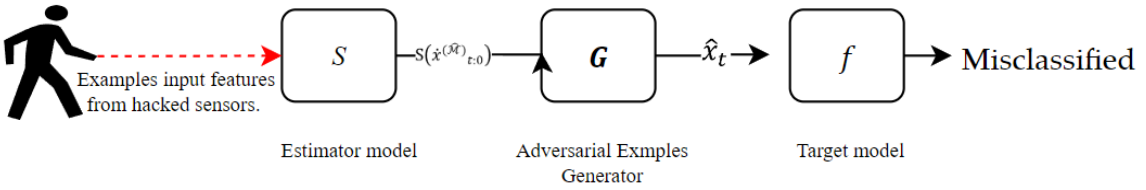
The attacker generates perturbations based only on the features from the hacked sensors. That is, the attacker uses $\dot{x}_t$. We denote $\dot{x}_{0:t} = (\dot{x}_0, \dot{x}_1, \ldots, \dot{x}_t)$. We define a function $G(\dot{x}_{0:t})$ whose inputs are the features monitored by the attacker and whose outputs are the generated perturbations. By adding perturbations generated by the generator $G(\dot{x}_{0:t})$, the features that include the attacks become $x'_t = x_t + B \times G(\dot{x}_{0:t})$.

In this study, we assume that the attacker has enough information about the target model and some knowledge of the sensors. The attacker can obtain the same model if the target uses an open model. Even if the model is not open, the information on the model can be extracted by conducting a model extraction attack [90], which steals the architecture, parameters, and hyperparameters of the target by monitoring the model's output if the attacker can use the model. This study assumes the case after obtaining accurate information on the target model. However, the stolen model may include estimation errors. The demonstration of the attacks in the case that the information of the target model is inaccurate is one of our future works.

On the other hand, knowledge of the sensors can be obtained through generally known knowledge. If it is generally known that values of a sensor correlate with the other sensors' values, attackers can use this knowledge. If the attackers can buy and use the same type of sensors, they can perform experiments to obtain the knowledge of the sensors. In this study, we model the knowledge of the sensors by an estimator $\hat{x}_t = S(\dot{x}_{t:0})$ whose input is the feature that can be monitored by the attacker and whose output is all features.



(a) Generated an adversarial attack using a generator model



(b) Process trained the generator.

Figure 3.2: Overview of architecture generated an adversarial example in multimodal sensors

3.3.2 Generation of Attack

In this study, the attack is generated using the generator $G(\dot{x}_{0:t})$, as shown in Figure 3.2a.

The generator $G(\dot{x}_{0:t})$ is trained in advance. The attacker can monitor the features of the target from the hacked sensors. We denote the dataset monitored from the hacked sensors as M. The attacker also has some knowledge of the other sensors and can estimate the values of the other sensors using the estimator $S(\dot{x}_{t:0})$, although the estimation may be inaccurate. Using the dataset M and estimator $S(\dot{x}_{t:0})$, the attacker can generate the dataset that can be used to train the generator. Hereafter, we denote the generated training data as $\hat{M}$. Each element of $\hat{M}$ can be generated by:

$$\hat{x}_t = \mathrm{S}\left(\dot{x}^{(\mathcal{M})}_{t:0}\right), \tag{3.1}$$

where $\dot{x}^{(M)}_{0:t} = (\dot{x}^{(M)}_0, \dot{x}^{(M)}_1, \ldots, \dot{x}^{(M)}_t)$, and $\dot{x}^{(M)}_t$ is an element of data in the dataset M.

Figure 3.2b shows the process to train the generator using the dataset $\hat{M}$. When training the generator, the attacker has the information on the target model $f(\cdot)$. Using $f(\cdot)$, the attacker trains the generator by minimizing the following loss function:

$$L_{\text{adv}}^{f} = \mathbb{E}_{x_t}\left[l_f\left(f(\hat{x}_t + B \circ G(\hat{x}_{0:t})), C_t\right)\right] \tag{3.2}$$

where $\hat{x}_{0:t} = (\hat{x}_0, \hat{x}_1, \ldots, \hat{x}_t)$, $l_f(y, C)$ is the loss function of the target model when the output of the target model is y, the target class is C, and C_t is the target class at time t. By generating the perturbation using the generator trained to minimize this loss function, the features after the attack can be classified into the attacker's desired class.

3.4 Experiments

3.4.1 Target Scenario

Overview

In this scenario, we use as a target model a machine learning model that identifies human activities from three sensors. This model is used to recognize human activities for healthcare, smart-home environment, and so on. In this model, the user wears three sensor devices on the chest, left ankle, and right wrist. All three sensor devices have 3D accelerometers. Moreover, the sensor device on the chest has an ECG sensor, and the other sensor devices have 3D gyroscopes and 3D magnetometers. The sensor devices send their monitored values to the server with the machine learning model based on a DNN. The server recognizes the user's current activity by handling time-series data sent from the sensors.

In this experience, we focus on specific subjects as the target and generate perturbations so that the activities of the specific subjects are identified as the target classes, which are different from the ground-truth classes.

To generate the perturbations, we assume that the attacker has hacked one of the sensor devices, the ankle sensor. The attacker can access and change the sensor values of the ankle sensor but cannot access the values of the other sensors. By changing the sensor values sent to the server, the attacker attempts to change the activity recognized by the machine learning model.

In this study, we assume that the attackers use their knowledge to train the attack

generator. In this scenario, we simulate the attacker's knowledge using an estimator trained by a dataset without the target subjects. By changing the amount of dataset used to train the estimator, we simulate the various cases—from the case that the attacker has accurate knowledge to the case in which the attacker has inaccurate knowledge.

Dataset

We used an open dataset called the MHealth dataset [91]. This dataset includes 12 physical activities (standing, sitting, lying down, walking, climbing stairs, bending forward, lifting arms forward, knees, cycling, jogging, running, and jumping back and forth) for ten subjects. They used wearable sensor devices located on the subject's chest, right wrist, and left ankle, and recorded the sensor values with a sampling frequency of 50 Hz. Table 3.3 lists the sensors used in the dataset.

The Mhealth dataset includes the time-series of sensor values. From this dataset, we extract the data with a length of 500 used for training and validation by using a sliding window. The number of extracted data for each subject and each class is shown in Table 3.4

Among ten subjects, we used subjects 9 and 10 as the target subjects. The data from the other subjects were used to train the target model and the estimator, but the data from the target subjects were not used to train the target model and estimator. When training the generator, we used the data of the target subjects but only the features of the hacked sensors, assuming that the attacker can access the values of the hacked sensors of the target subjects. The values of the other sensors used to train the generator are obtained using the estimator.

Target Model

This dissertation uses a model based on the long short-term memory network (LSTM) architecture proposed for HAR [92]. Figure 3.3 shows the architecture of the target model used in this study.

Sensor	**Locate**	**Abbreviated**
Acceleration from the chest sensor (X axis)		Acx
Acceleration from the chest sensor (Y axis)		Acy
Acceleration from the chest sensor (Z axis)	On Chest	Acz
Electrocardiogram signal (lead 1)		EL1
Electrocardiogram signal (lead 2)		EL2
Acceleration from the left-ankle sensor (X axis)		Alax
Acceleration from the left-ankle sensor (Y axis)		Alay
Acceleration from the left-ankle sensor (Z axis)		Alaz
Gyro from the left-ankle sensor (X axis)		Glax
Gyro from the left-ankle sensor (Y axis)	On Ankle	Glay
Gyro from the left-ankle sensor (Z axis)		Glaz
Magnetometer from the left-ankle sensor (X axis)		Mlax
Magnetometer from the left-ankle sensor (Y axis)		Mlay
Magnetometer from the left-ankle sensor (Z axis)		Mlaz
Acceleration from the right-lower-arm sensor (X axis)		Arlax
Acceleration from the right-lower-arm sensor (Y axis)		Arly
Acceleration from the right-lower-arm sensor (Z axis)		Arlz
Gyro from the right-lower-arm sensor (X axis)		Grlax
Gyro from the right-lower-arm sensor (Y axis)	On Wrist	Grlay
Gyro from the right-lower-arm sensor (Z axis)		Grlaz
Magnetometer from the right-lower-arm sensor (X axis)		Mrlax
Magnetometer from the right-lower-arm sensor (Y axis)		Mrlay
Magnetometer from the right-lower-arm sensor (Z axis)		Mrlaz

Table 3.3: Description of Sensors and their Abbreviations.

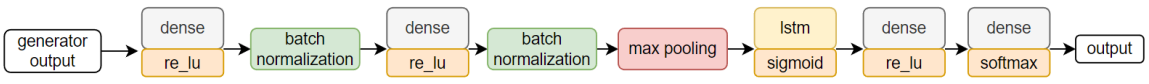
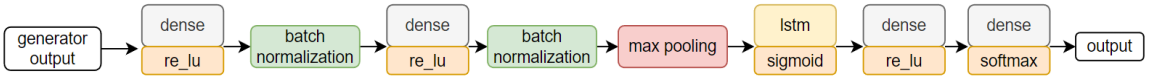


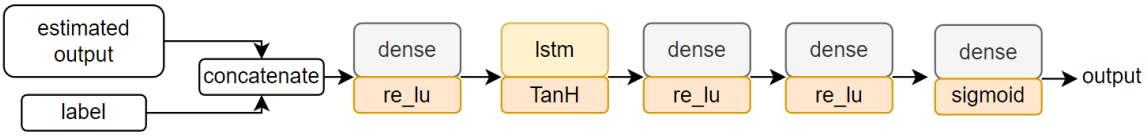
Figure 3.3: The architecture of the target model

Subject	**Standing**	**Lying Down**	**Walking**	**Climbing Stairs**	**Waist Bends Forward**	**Frontal Elevation of Arms**	**Knees Bending**	**Cycling**	**Jogging**	**Running**	**Jump Front and Back**
1	3072	3072	3072	3072	3072	3132	3278	3179	3072	3072	1024
2	3072	3072	3072	3072	3132	3132	3380	3430	3072	3072	1024
3	3072	3072	3072	3072	3132	3132	3266	3379	3175	3072	1024
4	3072	3072	3072	3072	3132	3132	3266	3379	3175	3072	1024
5	3072	3072	3072	3072	3132	3132	7265	7141	2784	3072	1024
6	3072	3072	3072	3072	2766	2894	2816	3072	3072	3072	1024
7	3072	3072	3072	3072	2766	2894	2816	3072	3072	3072	1024
8	3072	3072	3072	3072	2458	2765	2867	3072	3072	3072	1025
9	3072	3072	3072	3072	2458	2765	2867	3072	3072	3072	1025
10	3072	3072	3072	3072	2458	2765	2867	3072	3072	3072	1025

Table 3.4: Data that we used for each class and individual.



(a) Estimator model



(b) Discriminator model.

Figure 3.4: Estimation model architecture

We built the target model on top of TensorFlow and Keras and trained it using multiple NVIDIA Quadro RTX 5000. We trained the model to minimize the cross-entropy of the outputs and the corresponding labels in the training data using the Adam optimizer with a learning rate of 0.001, batch size of 32, and 100 epochs. Data from eight subjects were used to train the model. The data from the remaining subjects were used to evaluate the target model and the attack. To train the target model, we used time-series data generated by dividing the time-series data included in the training data into small sets of time-series data with lengths of 500 using the sliding-window technique.

Estimator

In this study, we constructed an estimator to simulate the attacker's knowledge. Estimator S estimates sensor values that are not obtained by the attacker from the values of the hacked sensors. In this study, we used the conditional-GAN training technique [93] to train estimator S, as shown in Figure 3.4.

In this technique, the discriminator D is introduced. The discriminator D distinguishes the output of the estimator from the training dataset. By training S to generate values that cannot be distinguished by discriminator D, we construct S to estimate the original values.

The discriminator D and the estimator S are trained alternatively. When training discriminator D, the parameters of discriminator D are updated to minimize the loss function, indicating the accuracy of the classification using the original values and the values

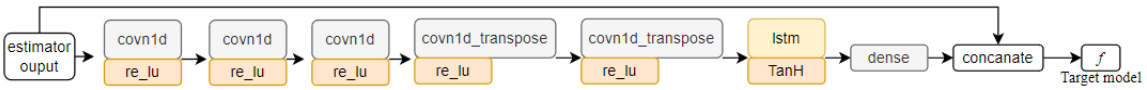


Figure 3.5: The architecture of the generator model.

generated by S as the training dataset. However, when training estimator S, we input the features of the hacked sensors of the training data to S, obtain the output from S indicating the estimated features, including the values of the other sensors, and use the output from S as the input for D. Finally, the output from D is obtained. Based on the output from D obtained by this process, the parameters of S are updated to minimize the same loss function of D by setting the target class of the generated values to the class for the original data.

We use LSTM in the estimator and discriminator to handle the time-series data. Figure 3.4 also shows the structures of the estimator and discriminator used in this demonstration. In the estimator, the values of the other sensors were estimated using the CNN–LSTM structure. Then, at the final layer, the estimator estimates the features of all the sensors, including the values of the hacked sensors, by concatenating the estimated sensor values and the values that can be monitored.

In this study, TensorFlow and Keras with multiple NVIDIA Quadro RTX 5000 were used. We trained the estimator and discriminator using 50 iterations with a learning rate of 0.002 and a batch size of 128. To train them, binary cross-entropy was used as the loss function. The data used to train the estimator and discriminator were generated by dividing the long time-series data using the sliding-window technique.

Generator

We use the generator based on 1D CNN and LSTM to handle the time-series data of the hacked sensors as inputs. Figure 3.5 shows the generator used in this study.

We trained the generator using features estimated by the estimator from the values of the hacked sensors of the target subjects. We used Adam optimization and set the batch size to 256, learning rate to 0.002, epochs to 1, and epsilon ϵ to 0.3. Similar to the target

model and estimator, we divided the time-series of features into time-series features with a sliding window size of 500 and used the divided time-series features to train the generator.

Before demonstrating the attacks, we investigated the properties of the target model by comparing it with a similar model using only one sensor device. In this comparison, we used precision and recall as metrics. Precision and recall are defined as:

$$\text{Precision} = \frac{tp}{tp + fp} \tag{3.3}$$

$$\text{Recall} = \frac{tp}{tp + fn} \tag{3.4}$$

where true positive *(tp)* is the number of data that can be classified correctly, false positive *(fp)* is the number of data that are classified into a class but whose correct class is different, and false negative *(fn)* is the number of data not classified into a class but whose correct class is the class.

Table 3.5 lists the precision and recall values for each class. This table shows that the model using only a single sensor device cannot recognize some classes. For example, the chest sensor cannot distinguish between standing and sitting states. The recall result shows that the wrist sensor cannot accurately recognize the walking class. The ankle sensor has high precision and recall compared with other sensors. However, even the ankle sensors achieve only 88% recall in the standing class. In contrast, the target model using all three sensors can recognize any class. In other words, multiple sensor devices are required for HAR.

In consideration of the limited space available on the page, the "Ground-Truth Class (GTC)" in our table has been effectively abbreviated to ensure conciseness without compromising the clarity of information. The categories are now represented as follows: Standing (St), Sitting (Si), Lying Down (LD), Walking (Wk), Climbing Stairs (CS), Waist Bends Forward (WBF), Frontal Elevation of Arms (FEA), Knees Bending (KB), Cycling (Cy), Jogging (Jg), Running (Rn), and Jump Front and Back (JFB).

We also investigate the impact of each sensor on the classification results. We use an

GTC	Precision				Recall			
	Target Model	**Wrist**	**Ankle**	**Chest**	**Target Model**	**Wrist**	**Ankle**	**Chest**
St	100	96	96	41	100	100	88	48
Si	100	99	98	54	100	96	97	54
LD	100	96	99	100	100	98	99	100
Wk	100	94	99	99	100	53	99	98
CS	99	79	98	98	100	97	99	100
WBF	100	83	96	100	100	94	99	94
FEA	98	100	89	87	100	99	97	75
KB	100	76	99	100	96	80	97	66
Cy	96	98	98	76	100	100	98	98
Jg	98	99	98	87	93	100	93	94
Ru	94	97	93	92	97	99	98	89
JFB	95	97	92	100	97	89	94	94

Table 3.5: Complete results of the three other models with the target model using different training data.

integrated gradient [94]. The integrated gradient is a method for evaluating the impact of each feature on the results of a machine learning model.

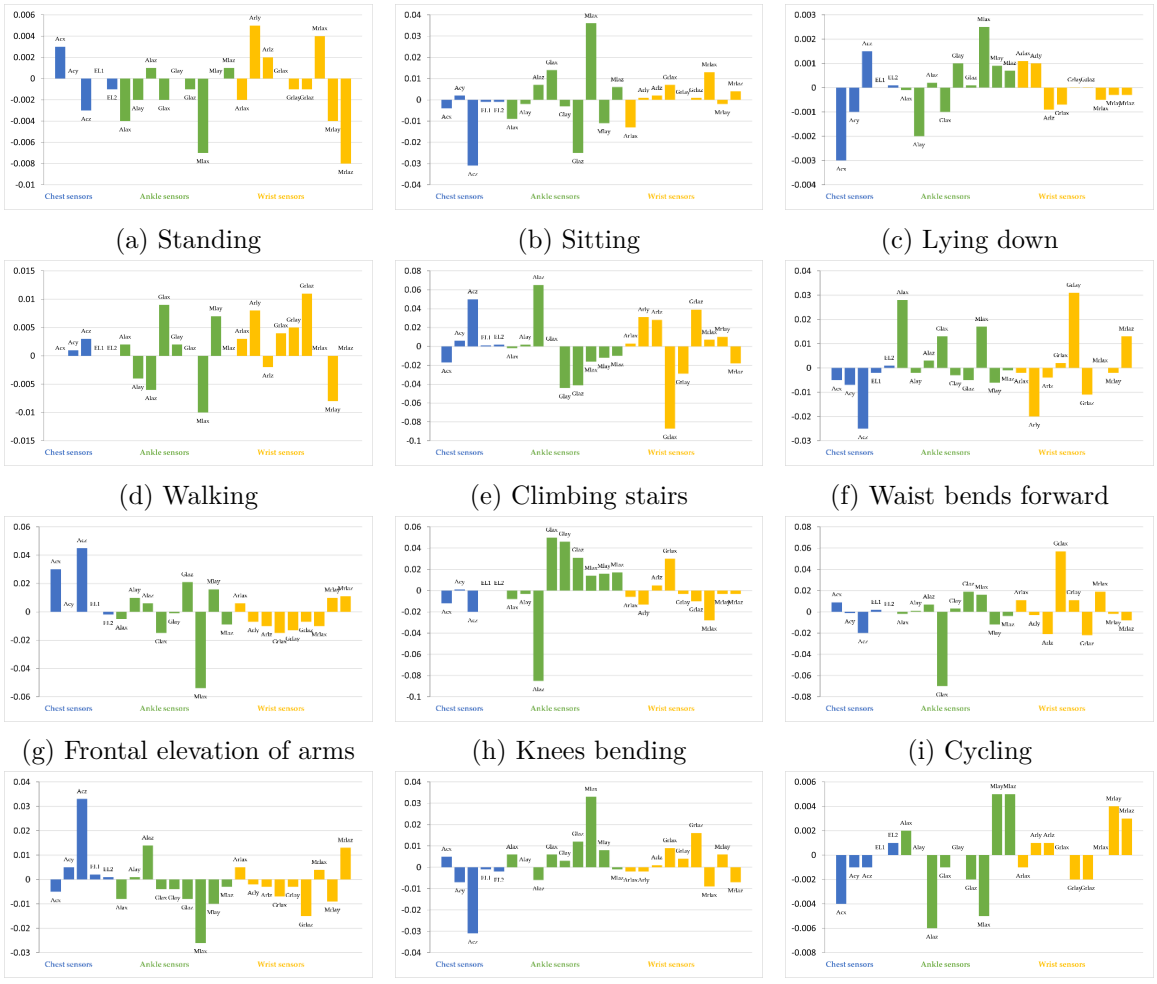
The integrated gradient of the ith feature of the input x on the class j is defined as

$$\text{IntegratedGrads}_{i,j}(x) := (x^{(i)} - x'^{(i)}) \times \int_{\alpha=0}^{1} \frac{\partial f_j(x + \alpha(x - x'))}{\partial x^{(i)}} \, d\alpha \tag{3.5}$$

where x' represents the baseline input, $x^{(i)}$ is the ith feature of x, and α is the interpolation constant. The features whose integrated gradient is far from zero have a significant impact on the output of the model. We calculate the integrated gradient for data of the target subject and calculate the average of them for each class.

Figure 3.6 shows the integrated gradient for 12 classes. The vertical axis in Figure 3.6 is the integrated gradient for each feature. A large positive integrated gradient means that the feature has a strong positive correlation to the class, and a large negative integrated gradient means that the feature has a strong negative correlation. If the integrated gradient is close to 0, the corresponding feature does not contribute to the classification.

Figure 3.6a indicates that Mlax from the ankle sensor and Arly, Mrlax, and Mrlaz from the wrist sensor have a strong correlation to the classification into the Standing class and



(a) Standing (b) Sitting (c) Lying down

(d) Walking (e) Climbing stairs (f) Waist bends forward

(g) Frontal elevation of arms (h) Knees bending (i) Cycling

(j) Jogging (k) Running (l) Jump front and back

Figure 3.6: Impact result of input feature sensors for each class in the target model using integrated gradients (IG).

Ground-Truth	**Five Subjects**	**Three Subjects**
Standing	25.57	28.18
Sitting	19.11	26.31
Lying down	22.08	26.01
Walking	21.75	25.38
Climbing stairs	21.74	25.50
Waist bends forward	28.69	35.91
Frontal elevation of arms	22.30	28.54
Knees bending	22.24	33.81
Cycling	20.06	26.04
Jogging	22.71	35.52
Running	20.00	35.61
Jump front and back	21.76	34.13

Table 3.6: Results of the MSE on the model estimator using five and three subjects.

have a large impact on the classification results. Similarly, the other figures in Figure 3.6 indicate the features that contribute to the classification. From this figure, multiple sensors contribute to classification into any classes in the target model. Namely, our target model identifies all classes based not only on specific sensor devices, but also on multiple sensor devices.

3.4.2 Property of the Estimator.

In Table 3.6, we show the results of the estimator trained using five and three subjects. In this table, we evaluate the accuracy of the estimator using the mean square error (MSE) as a metric. The MSE is defined as

$$\text{MSE} = \frac{\sum_{t=1}^{n} \sum_{i \in \{i|o_i=1\}} (x_{t,i} - \hat{x}_{t,i})^2}{n}, \tag{3.6}$$

where $x_{t,i}$ and $\hat{x}_{t,i}$ are the actual and estimated values of the ith feature at time t, and n is the length of the validation data. The smaller the MSE value is, the more accurate the prediction results are.

In this study, we trained the estimator using three and five subjects and then evaluated the accuracy of the data using data from two subjects that were not included in the training

dataset for the target model and the estimator. Table 3.6 shows that the evaluation errors increased as the number of subjects used to train the estimator decreased. In the remainder of this subsection 3.4.2, we investigate whether attackers with these estimators can succeed in the attack.

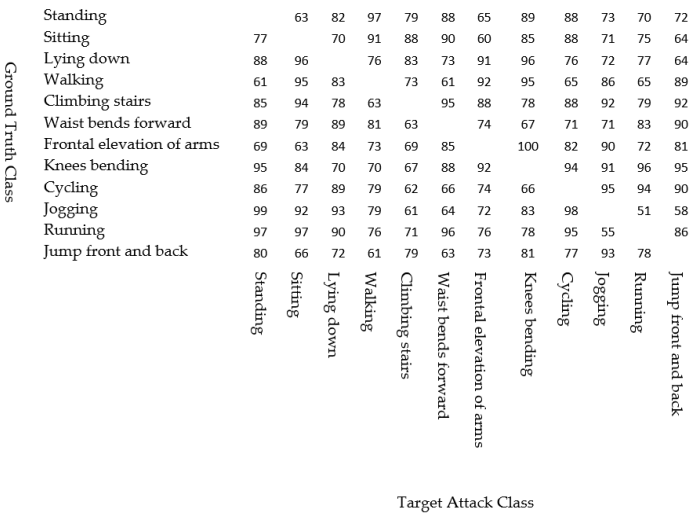
3.4.3 Demonstration of the Attack.

In this subsection 3.4.3, we describe the attacks. To evaluate the generated attacks, we introduce a metric called the *attack success ratio*, which is defined as $\frac{N^{\text{success}}}{N^{\text{attack}}}$, where N^{attack} is the total number of time slots, including the attacks, and N^{success} is the number of time slots in which the results of the target model are the attacker's desired classes. Figure 3.7 shows the results. The rest of this subsection 3.4.3 discusses the results.

Case That the Attacker Has Full Knowledge of All Sensors

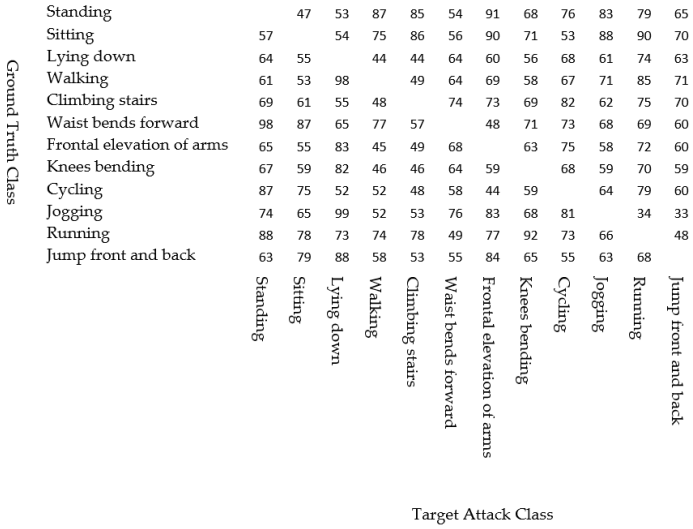
Before discussing the results of the impact of the attacker's knowledge, we first investigate the case in which the attacker has sufficient knowledge. In this case, we use the actual values of all the sensors to train the generator instead of using the estimated values. Note that even in this case, the attacker does not have the values of the other sensors during attacks but can monitor only the features from the hacked sensors.

Figure 3.7a shows the results that the attack success ratio depends on the ground truth and target classes. For example, all attacks from frontal elevation of arms to knee bending succeeded, whereas the attack success ratio of attacks from jogging to running was low. However, even in the case with the lowest success ratio, more than half of the attacks succeeded. That is, the attacks succeeded by changing the values of the ankle sensors, although the other sensors also have a large impact on the classification results. We discuss the property of the generated attacks in Subsection 3.4.2



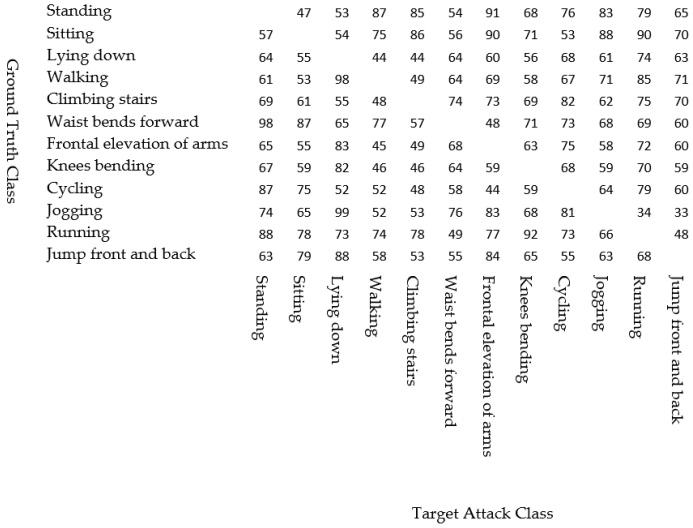
	Standing	Sitting	Lying down	Walking	Climbing stairs	Waist bends forward	Frontal elevation of arms	Knees bending	Cycling	Jogging	Running	Jump front and back
Standing		63	82	97	79	88	65	89	88	73	70	72
Sitting	77		70	91	88	90	60	85	88	71	75	64
Lying down	88	96		76	83	73	91	96	76	72	77	64
Walking	61	95	83		73	61	92	95	65	86	65	89
Climbing stairs	85	94	78	63		95	88	78	88	92	79	92
Waist bends forward	89	79	89	81	63		74	67	71	71	83	90
Frontal elevation of arms	69	63	84	73	69	85		100	82	90	72	81
Knees bending	95	84	70	70	67	88	92		94	91	96	95
Cycling	86	77	89	79	62	66	74	66		95	94	90
Jogging	99	92	93	79	61	64	72	83	98		51	58
Running	97	97	90	76	71	96	76	78	95	55		86
Jump front and back	80	66	72	61	79	63	73	81	77	93	78	

(a) case with full knowledge



	Standing	Sitting	Lying down	Walking	Climbing stairs	Waist bends forward	Frontal elevation of arms	Knees bending	Cycling	Jogging	Running	Jump front and back
Standing		47	53	87	85	54	91	68	76	83	79	65
Sitting	57		54	75	86	56	90	71	53	88	90	70
Lying down	64	55		44	44	64	60	56	68	61	74	63
Walking	61	53	98		49	64	69	58	67	71	85	71
Climbing stairs	69	61	55	48		74	73	69	82	62	75	70
Waist bends forward	98	87	65	77	57		48	71	73	68	69	60
Frontal elevation of arms	65	55	83	45	49	68		63	75	58	72	60
Knees bending	67	59	82	46	46	64	59		68	59	70	59
Cycling	87	75	52	52	48	58	44	59		64	79	60
Jogging	74	65	99	52	53	76	83	68	81		34	33
Running	88	78	73	74	78	49	77	92	73	66		48
Jump front and back	63	79	88	58	53	55	84	65	55	63	68	

(b) case with estimator trained by five subjects



	Standing	Sitting	Lying down	Walking	Climbing stairs	Waist bends forward	Frontal elevation of arms	Knees bending	Cycling	Jogging	Running	Jump front and back
Standing		47	53	87	85	54	91	68	76	83	79	65
Sitting	57		54	75	86	56	90	71	53	88	90	70
Lying down	64	55		44	44	64	60	56	68	61	74	63
Walking	61	53	98		49	64	69	58	67	71	85	71
Climbing stairs	69	61	55	48		74	73	69	82	62	75	70
Waist bends forward	98	87	65	77	57		48	71	73	68	69	60
Frontal elevation of arms	65	55	83	45	49	68		63	75	58	72	60
Knees bending	67	59	82	46	46	64	59		68	59	70	59
Cycling	87	75	52	52	48	58	44	59		64	79	60
Jogging	74	65	99	52	53	76	83	68	81		34	33
Running	88	78	73	74	78	49	77	92	73	66		48
Jump front and back	63	79	88	58	53	55	84	65	55	63	68	

(c) case with estimator trained by three subjects

Figure 3.7: Results of attacks on ankle sensors using full knowledge, results of success attack rate using five subjects, and results of success attack rate on ankle group sensor using three subjects.

The Impact on the Attacker's Knowledge

Figures 3.7b, 3.7c show the attack success ratio for the cases in which the attacker has the estimator trained by five and three subjects, respectively. As discussed above, as the number of subjects used to train the estimator decreases, the estimation errors increase. Consequently, the attack success ratio also decreases. Figure 3.7c also indicates that the attacks for some classes succeeded even if the attacker did not have accurate information on the other sensors. For example, the attacks from frontal elevation of arms to walking succeeded with a high attack success ratio, while most attacks from frontal elevation of arms to jogging or jumping front and back failed. We discuss the cause of these differences in the results in Subsection 3.4.4

3.4.4 Property of the Generated Attacks

In this subsection 3.4.4, we discuss the properties of the generated attacks. We calculate the IGs of the generated attacks. Figure 3.8 shows examples of the average integrated gradients of the attack generated for data whose ground-truth class is frontal elevation of arms.

In Figure 3.8, compared with Figure 3.6, the features that include attacks that have a large impact on the classification are very different from the data without attacks. This is because the generator does not generate the perturbation to make the input of the model similar to the normal data whose class is the target class. However, the generator generates the perturbations to minimize the loss function of the target model. As a result, the features are very different from the original data but are classified into the target class by the target model.

Figure 3.8 also shows that the features of the sensors that are not hacked have a large impact on the classification results in some cases. In this case, by changing the features from the hacked sensor, the attacker moves the features to a location corresponding to the target class in the feature space. Among the changed features, the features from the other sensors contribute to the classification of the target class. However, to succeed in this type of attack, the attacker must know the impact of the features from the other sensors on the

(a) Ground Truth: Frontal elevation of arms, Target: Walking (case with full knowledge) Target: Jogging (case with full knowledge)

(b) Ground Truth: Frontal elevation of arms,Target: Jogging (case with full knowledge)

(c) Ground Truth: Frontal elevation of arms, Target: Jump front and back (case with full knowledge)

(d) Ground Truth: Frontal elevation of arms, Target: Walking (case with estimator trained by three subjects)

Figure 3.8: Integrated gradients of the inputs that include attacks.

classification results. That is, accurate information from other sensors is required.

However, as shown in Figure 3.7, even if the attacker's knowledge is inaccurate, attacks on certain classes succeed. One example is the attack from frontal elevation of arms to walking. Figure 8d shows the average IGs of the data with attacks generated in the case with the estimator trained with three subjects. Figure 3.8d indicates that the features from the ankle sensor have the largest impact on the classification results; therefore, this attack succeeded even if the attacker does not have accurate knowledge of the other sensors. In this case, the attacks cause the classification into the target class by increasing the contributions of the features of the hacked sensors for the target class. The attacker can determine and change the values of the features of the hacked sensors. That is, the attacker can accurately calculate the contributions of the features of the hacked sensors and change the features to increase the contributions. As a result, this attack succeeds even if the attacker does not

have sufficient information from other sensors.

3.5 Discussion

In this study, we demonstrate that attacks that cause misclassification in target models are possible even if the attacker hacked a part of the sensors. In particular, if the attacker has sufficient knowledge of the other sensors, the attack succeeds with a high probability, although the attacker cannot monitor the current values of the other sensors. We also demonstrate that the attacks succeed in some cases, even if the attacker does not have sufficient knowledge of other sensors.

In our experiment, we focus on one model as a target model. However, our approach is not based on any assumptions about the target model. Thus, this kind of attack is possible in the other models, though demonstration using the other models and a different dataset is one of our future research topics.

In this study, we assume that the attacker has some knowledge of the legitimate sensors and we simulated this knowledge by using the estimator. By using the estimator trained by a limited amount of dataset, we simulated the case that the attacker's knowledge is inaccurate. In the actual situation, the attackers may obtain knowledge of the legitimate sensors by using generally known knowledge of the sensors or performing experiments using the same sensor by themselves. However, if the target has properties that are quite different from the knowledge obtained by the attackers, the attacks become more difficult, the evaluation of the attacks in the case that the properties of the target are quite different from the attackers' knowledge.

In this study, we also assume that the attacker has enough information on the target models. The attacker, however, may have only insufficient information on the target model. Especially, the target model may become different if it is updated. The adversarial examples in the case that the attacker does not have information on the target models have also been discussed [95], and the attacks combining the approach in this study with such methods are possible. Demonstrating such attacks is one of our future research topics.

Though we need further research to demonstrate the attack in other cases, the results of this study indicate that the service provider using a machine learning model based on multiple sensors should consider the case in which some of the sensors may be hacked by the attacker. By considering these attacks, we may be able to construct robust models. One of the approaches to constructing robust models is to use the adversarial training, considering the attacks [15]. However, the robustness of adversarial training against such attacks has not yet been discussed, and further research is required. Another approach against an attack from a part of the hacked sensor is to utilize the properties of this attack. Because the attacker cannot access the other sensors, the generated signals may include some inconsistency between the signals from the other sensors. These countermeasures will be a future research topic.

In this study, we investigated the properties of generated attacks. The results indicate that the attacker does not need to generate input signals that are similar to the actual features of the target class. However, these results do not indicate that the signals generated by hacking a small number of sensors are different from the actual features. By training the generator, considering the difference from the actual features of the target class, it may be possible to generate attacks that are difficult to detect based on the difference from the normal features of the target class. Therefore, in Chapter 4, we aim to clarify the properties of attacks that cannot be avoided by attackers.

3.6 Conclusions

In this chapter, we discussed the possibility of attacks on DNN models by hacking a small number of sensors. In this scenario, the attacker first hacks a few sensors; then, the attacker can obtain the values of the hacked sensors and change them, but the attacker cannot obtain and change the values of the other sensors.

In this study, we introduced a generator that generates adversarial examples when a small number of sensor devices are hacked. The generator uses the values from the hacked sensors as inputs and generates perturbations so that the features, including the

perturbation, are classified into the target class by the target model.

We demonstrated the attack using an open dataset for HAR based on three sensor devices located on the chest, wrist, and ankle of the subjects. We then clarified that the attacker can change the output of the target model by hacking only one of the three devices.

The next chapter focuses on the properties of attacks, such as countermeasures against attacks.

Chapter 4

Detection of Sensors Used for Adversarial Examples Against Machine Learning Models

4.1 Introduction

The integration of machine learning (ML) with multiple sensors has revolutionized various critical domains, such as healthcare [10], autonomous vehicles [96], and other fields [97]. In these systems, observations from sensor devices are collected via the Internet or a network managed by the service provider. Then, ML models recognize the current situation using the collected observations as their input. The utilization of observations from multiple sensor devices in these systems has shown to significantly enhance recognition accuracy. For instance, Namazi et al. developed a method to detect and track continuous objects surrounding vehicles, locate objects in front of the vehicle, and mitigate occlusion issues to provide more precise readings [11]. Similarly, multiple systems, Zhou et al. [98] and Yuan et al. [99] proposed systems to recognize human activities in low-light indoor conditions by using multiple sensor devices.

However, as systems using ML models have become more widely used, these models

have also become the target of attacks. Adversarial examples (AEs) are inputs designed by an adversary that causes a ML system to generate incorrect outputs. Finlayson et al. showed that an adversary can exploit the vulnerability of ML models by creating AEs in critical domains, such as medical [12]. These AEs contain subtle changes that are invisible to humans but can mislead the model's predictions. This poses a significant threat to the systems based on ML models, particularly in sensitive fields.

Considering the systems using multiple sensors, some sensors may be vulnerable and can be used to generate AEs. The data manipulation of sensor data by compromising the software in the sensor device was demonstrated [13]. Monjur et al. have been demonstrated that data manipulation is also possible by modifying the hardware if the attacker can physically access the sensor device [14]. We also demonstrated that the attacker can change the output of the ML models using multiple sensors if the attacker can manipulate the values from a part of sensors [48]. That is, manipulation on all sensor values is not necessary for the attacks to change the output values of the ML models. We call this attack *sensor-based AEs*. However, it is difficult to protect all sensor devices, because the risk of the existence of the vulnerable sensor devices increases as the number of sensor devices increases. Therefore, we need a method to protect ML models even if a part of sensors are compromised by the attacker.

Many countermeasures to mitigate the risks posed by AEs have been proposed. One approach is to make the ML models robust against AEs. Adversarial training (AT) is one of the methods used to enhance the robustness of ML models [15]. In this method, ML models are iteratively trained using clean and adversarial examples generated for the current model. By utilizing the AEs, this method enhances the robustness of the model. However, the model trained by AT is only robust against the attack generated during the training and may be vulnerable to new attacks [16].

Another approach to countering AEs is to detect attacks. Especially in the case of sensor-based AEs, the detection of the sensors used in the sensor-based AEs is essential. By detecting the sensors used in the attack, we can check the sensors and replace them. Many methods to detect AEs have also been proposed [17–24]. Hendrycks and Gimpel

introduced detection methods for adversarial examples based on identifying implausible gradients [20, 21]. Metzen et al. proposed a method based on Lipschitz continuity to detect adversarial perturbations, which involve subtle changes applied to clean examples [23].

However, all existing detection methods only detect attacks and do not detect the features changed by the attacker because they aim to detect the inputs to be blocked. On the other hand, detecting the sensors used by the attacker is essential for systems with multiple sensors, as discussed above. In addition, existing detection methods can be avoided by modifying the AE generation process [25]. This is caused by the features used to detect AEs; the existing methods use features of the AEs, but an attacker with the knowledge of the detector can change the features. That is, the detection method should use the features of the AEs that cannot be avoided by an attacker.

We propose a method to detect the attacks and the sensors used by the attackers. In this method, we introduce a model called the feature-removable model (FRM) that allows us to select the features used as an input into the model [49]. We obtain the outputs of the FRM using all features and features from some of the sensors. If we find inconsistencies between the outputs, our method detects the attacks. Then we also detect the sensors the attacker uses by finding the sensors causing the inconsistency.

After detecting the sensors the attacker uses, we can check and replace them. Nevertheless, our FRM can also be used after the identification; we can obtain the output of the FRM without using the features from the detected sensors to avoid the impact of the attacks, though the accuracy of the model might decrease compared with the case that we can use all features.

To the best of our knowledge, we are the first to propose a method that can detect the sensors used in sensor-based AEs. Our method is based on the features of the sensor-based AEs that the attacker cannot avoid; the output of the ML model is altered when the values from the sensors used by the attacker are incorporated.

After detecting the sensors the attacker uses, we can use our FRM to keep the system work; we can obtain the output of the FRM without using the features from the detected sensors to avoid the impact of the attacks.

However, such reactive defense method has limitations. If some critical sensors that are necessary to distinguish required states are compromised by the attacker, we cannot obtain the suitable output even if we use the FRM without using the features from the detected sensors. As a result, we cannot keep the system work.

In this thesis, we discuss a strategy to make the system robust against sensor-based AEs proactively. A system with enough redundancy can work after removing the features from the sensors used in the sensor-based AEs. That is, we need a metric to check if the system has enough redundancy. In this study, we define groups of sensors that might be compromised by the same attacker, and propose a metric called *criticality* that indicates how important each group of sensors is for classification between two classes [50]. Based on the criticality, we can make the system robust against sensor-based AEs by interactively adding sensors so as to decrease the criticality of any sensors for the classes that must be distinguished.

The structure of this chapter is as follows: Section 4.2 offers an overview of existing defense methods against adversarial examples (AEs). In Section 4.3, we provide a comprehensive discussion and definition of the sensor-based adversarial attacks that are the focus of this chapter. Section 4.4 outlines our newly proposed framework, which is designed to defend against attacks using sensor-based adversarial examples. Section 4.5 assesses the effectiveness of our proposed countermeasures through a series of experiments. Section 4.6 explores the importance and impact of sensor vulnerabilities in relation to these attacks. Section 4.7 discussion toward robust systems against sensor-based AEs Finally, Section 4.8 wraps up the chapter and looks ahead to future research possibilities in this field.

4.2 Related Work

Many countermeasures to mitigate the risks posed by AEs have been proposed. One of the approaches is the AT. AT is a technique used to make the ML model robust against AEs by incorporating AEs during the training process. AT minimizes the loss for clean inputs and the generated AEs. Ensemble Adversarial Training (EAT) [16] is a method that improves

the robustness of AT. In this method, multiple models are trained at the same time, and AEs generated for a model is used to train the other models.

However, AT has a limitation. It relies on specific attacks generated during training, which hampers its ability to defend against new and unseen attacks [100, 101].

Another approach to mitigating the risks posed by AEs is to detect AEs. Hendricks and Gimpel introduced three detection methods, referred to as the H&G detection methods [20, 21]. They are based on 1) the coefficients in a PCA-whitened input, 2) the Softmax distributions, and 3) the reconstruction errors of the inputs reconstructed by an auxiliary decoder.

Another approach is based on ML models to detect AEs. Gong et al. [17] proposed a method based on a binary classifier trained independently from the original model. They trained the binary classifier to output 0 for the clean data and 1 for the AEs. Metzen et al. [23] also proposed a method based on binary classifiers. They constructed binary classifiers as subnetworks connected to each layer of the original main model. Grosse et al. augmented an ML model with an additional output, in which the model is trained to classify adversarial inputs [18]. Hosseini et al. trained an ML model to smoothly decrease its confidence in the original label and instead predicted that the input is "invalid" when it is more perturbed [22]. Miller et al. proposed a detection method for anomaly detection, which utilizes a separate unsupervised learning procedure to detect AEs inputs [24].

In this thesis, we discuss a method to detect sensors used in sensor-based AEs. This method should satisfy the following requirements. 1) It should detect AEs accurately. 2) It should detect sensors used in sensor-based AEs. 3) It should utilize the features of sensor-based AEs that cannot be avoided by an attacker. If an attacker can avoid the features used by the detector, AEs that cannot be detected are possible.

Table 4.1 shows the existing method to detect AEs and includes whether each method meets the requirements. As shown in Table 4.1, none of the existing methods can detect sensors used in sensor-based AEs, because they did not focus on the sensor-based AEs. Furthermore, none of the existing methods utilizes features that cannot be avoided by the attacker. If an attacker can change the features used by a detector, they can generate AEs

Methods	Overview	Detection of attacks	Detection of sensors used in sensor-based AEs	Sure utilization of the unavoidable features of AEs
H & G [20, 21]	Detect AEs based on 1) the coefficients in a PCA-whitened input, 2) the Softmax distributions and 3) the reconstruction errors.	Yes	No	No
Metzen et al. [23]	Detect AEs by binary detector sub-networks using the outputs of each layer of the original model as inputs.	Yes	No	No
Gong et al. [17]	Detect AEs by a binary classifier that is trained independently from the original model.	Yes	No	No
Grosse et al. [18]	Augments a ML model with an additional output, in which the model is trained to classify adversarial inputs.	Yes	No	No
Hosseini et al. [22]	Train a ML model so that it outputs lower confidence on the original label and instead predicts that the input is "invalid", as the input is more perturbed	Yes	No	No
Miller et al. [24]	Detect AEs using unsupervised anomaly detectors.	Yes	No''	No
This study	Detect AEs from a part of sensors and identify the sensors used by the attacker by finding the inconsistency between the results using all features and without using features from a part of sensors	Yes	Yes	Yes

Table 4.1: Comparison of different AE detection methods.

that cannot be detected. Carlini et al. evaluated AE detection methods in a white-box setting and demonstrated that AEs capable of evading detection can be generated for all the methods evaluated [25].

In this thesis, we propose a method to detect the sensors used in sensor-based AEs that satisfies all of the above requirements. Our method utilizes the features of the attacks; that the results of the classification are change by using the features from the sensors used in the attack. Such features of the attacks cannot be avoided by the attacker.

4.3 Sensor-based Adversarial Examples.

In this thesis, we focus on the system that gathers values from multiple sensors and performs classification tasks based on ML models. We call this system as the target system.

We model the target system as the function $f(x_{0:t})$ where $x_{0:t} = (x_0, x_1, \ldots, x_t)$ is the input of the target system built from the sensor data received from time 0 to time t and x_t is the vector corresponding to the sensor values at time t. We refer to the j-th element of the model's output as $f_j(x_{0:t})$, and $f_j(x_{0:t})$ denotes the probability that the state at time t is classified into the i-th class. $f(x_{0:t})$ represents the classification outcome at time t.

The vector x_t is constructed of the values from multiple sensors. The values of the compromised sensors can be monitored and modified by the attacker. The information of the compromised sensors are represented by the vector $\mathbf{B}$, which is defined as $\mathbf{B} = (b_1, b_2, \ldots, b_m)$; $b_i = 1$ if the i-th value is from the compromised sensor. The sensor values that the attacker can monitor and modify at times t are given by $\dot{x}_t = \mathbf{B} \circ x_t$, where $\circ$ stands for the element-wise product.

Based on the sensor values of the compromised sensors, the attacker creates perturbation. The sensor values including the attacks become $x'_t = x_t + \mathbf{B} \circ G(\dot{x}_{0:t})$ where $G(\dot{x}_{0:t})$ is the attack generator and $\dot{x}_{0:t} = (\dot{x}_0, \dot{x}_1, \ldots, \dot{x}_t)$. The attacker can generate the attacks by training $G(\dot{x}_{0:t})$ so that the output becomes the class the attacker wants. In this study, we assume that an attacker has sufficient information about the target system.

On the other hand, the victim who manages the target system does not know the

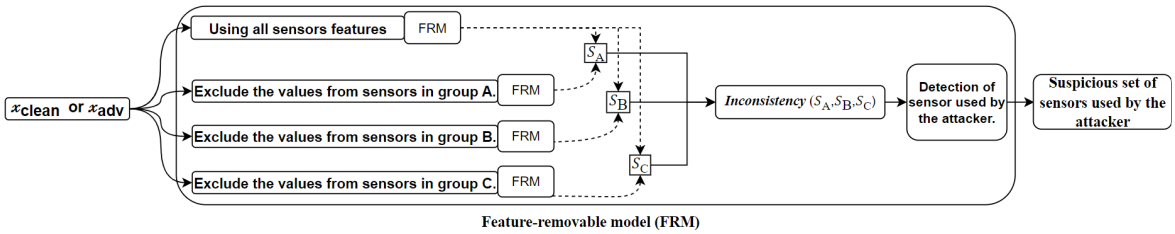


Figure 4.1: Illustration of countermeasure stage flow.

compromised sensors but knows the set of sensors with the same risk. For example, sensors connected to the same computer might be compromised at the same time if the computer is vulnerable. The same type of sensors may have the same vulnerability. In this study, we denote the set of risks by $\mathcal{R}$ and the set of sensors with risk $r \in \mathcal{R}$ by S_r. We assume that the victim has the information of S_r for all $r \in \mathcal{R}$.

4.4 Framework Against Sensor-based Adversarial Examples.

The attacker aims to change the classification results by changing the values of the compromised sensors. That is, the classification results obtained from compromised sensors are different from those obtained without using compromised sensors. Our method detects not only sensor-based AEs but also the sensors used in the attack by detecting these inconsistencies.

Figure 4.1 shows an overview of our method. In our method, we introduce a model called the feature-removed model (FRM), which allows us to select the features used for classification. By using the FRM, we can obtain multiple results by changing the features used for classification. Then, we detect attacks and sensors used in the attack by comparing the results and identifying any inconsistencies

4.4.1 Feature Removable Model

In this subsection, we propose a model called the FRM. The FRM is a model that allows us to exclude specific features. The FRM can be constructed by modifying the first and

last layers of the original model.

At the first layer, we add a function to exclude the features that are marked to be excluded. The eliminated features are set to 0. Then we apply the scaling based on the dropout [102]. The output of the first layer after scaling is obtained by

$$o_{1,i} = a\left(\frac{N^{\text{all}}}{N^{\text{selected}}}\sum_{k}(w_{0,k,i}o_{0,k}) + b_{1,i}\right) \tag{4.1}$$

where $o_{i,j}$ is the value of the j-th node at the i-th layer, $w_{0,k,i}$ and $b_{1,i}$ are the weight and bias, and $a(\cdot)$ is the activation function. The number of all features is N^{all} and the number of selected features is N^{selected}. By this scaling, we have a similar number of activated nodes to the case of using all features even if we exclude some features.

In the final layer, we employed the sigmoid activation function to allow the outputs for multiple classes to be large. If essential features are excluded, the FRM may be unable to identify the class. By allowing large outputs for multiple classes, the FRM can handle such cases by outputting large probabilities for all possible classes.

We train the FRM so that the outputted probabilities for all possible classes become large even if we exclude some features. In this thesis, we train the FRM by selecting the features to be excluded randomly. During the training, we use the following loss function.

$$L^{\text{"removed-feature"}}(Y,T) = -\sum_{i} w(t_i)(t_i \log y_i + (1-t_i)\log(1-y_i)) \tag{4.2}$$

where Y is the model's output, t_i is the i-th element of T, y_i is the i-th element of Y, and T is the training label. If the training label is i, t_i is set to 1. If not, 0. $w(t_i)$ is defined as the weight for t_i. We set $w(0) \ll w(1)$ to include the training label in the output. By using this loss function, we set a large penalty for the case that the actual class is not included in the output classes.

4.4.2 Detection of Attacks and Identification of Compromised Sensors

Our method detects the sensor-based AE attacks and the sensors used in the attack by comparing the results of the FRM. If the outputted probability for a class obtained by using all features is high but the corresponding probability obtained by excluding values from a set of sensors, the sensors are suspicious. Based on such inconsistencies, our method detects sensors used in the sensor-based AEs.

Algorithm 1 Detection of sensors used to generate AEs

Require: α (threshold for prediction), β (threshold for inconsistency), FF (features), *model* (trained feature removable model), SS (set of sensors), S_r (set of sensors with the risk r), RR (set of risks)
Ensure: *DetectedSensors*
1: $P_{all} \leftarrow model.predict\_using\_selected\_sensors(F, S)$
2: $indices \leftarrow extract\_indices\_exceeding\_threshold(P_{all}, \alpha)$
3: $P'_{all} \leftarrow extract\_result\_for\_indices(P_{all}, indices)$
4: **for** each r in RR **do**
5: $\quad P_{part} \leftarrow model.predict\_using\_selected\_sensors(F, S \backslash R_s)$
6: $\quad P'_{part} \leftarrow extract\_result\_for\_indices(P_{part}, indices)$
7: $\quad$ **if** $|P'_{all} - P'_{part}| > \beta$ **then**
8: $\quad\quad$ **for** each s in S_r **do**
9: $\quad\quad\quad DetectedSensor.Add(s)$
10: $\quad\quad$ **end for**
11: $\quad$ **end if**
12: **end for**
13: **return** *DetectedSensor*

Algorithm 1 shows the steps for detecting sensor-based AEs and the suspicious sensors used in the AEs. In these steps, we focus on the classes whose probabilities in the prediction results using all features exceed the threshold (α). Such classes are extracted as indices at Line 2 in **Algorithm 1**. Then, we obtain the results by excluding the values from the sensors with the risk ($r \in R$) and compare the results of the classes in the indices. If the difference exceeds a threshold (β) (Lines 7-11 in **Algorithm 1**), we detect the attacks and mark the sensors with the risk (r) as suspicious.

Algorithm 1 returns the set of suspicious sensors. Thus, if no suspicious sensors are found, we regard the current input as clean. But if at least one sensor is included in the

set of suspicious sensors, we detect sensor-based AEs.

4.5 Experiment.

4.5.1 Original Target Model, Dataset, and Attacks

In our experiment, we used a system that recognized human activity as the target system. In this system, three devices were mounted at three places on the chest, left ankle, and right wrist. All three devices had 3D accelerometers; the device on the chest had an ECG sensor, and the other devices had 3D gyroscopes and 3D magnetometers. Each device sends its monitored value to the server. The server recognizes the user's current activity by using the deep neural network (DNN) to handle time-series data sent from the devices.

Figure 4.2a depicts the DNN structure utilized in our experiment, which was inspired by the HAR model proposed by [92]. In this experiment, we use this structure as the original model. In this structure, we construct a vector by concatenating the sensor values obtained at each time slot and use it as an input. This model handles the continuous inputs by using an LSTM layer, and outputs the probabilities of the activities for the inputs.

From this original model, we constructed an FRM, modifying only the first and last layers, as detailed in Section 4.4. To train the FRM, we utilized the Adam optimizer with a learning rate of 0.001 and a batch size of 32 for 100 epochs. We used a custom binary cross-entropy loss function for training FRM with weight parameters. We set ww (0) to 1.0 and ww (1) to 20.0.

In this experiment, we use the MHealth dataset [91]. This dataset includes 12 physical activities (standing, sitting, lying down, walking, climbing stairs, bending forward, lifting arms forward, knees, cycling, jogging, running, and jumping back and forth) for ten subjects. In this experiment, we regard subjects 9 and 10 as the target subjects for whom the attacks are generated, and the data from the other subjects are used to train the FRM.

The Mhealth dataset includes time series of sensor values. We extract the data with a length of 500 from this dataset for training and validation by using a sliding window. Table 4.2 shows the number of data extracted for each subject and each class.

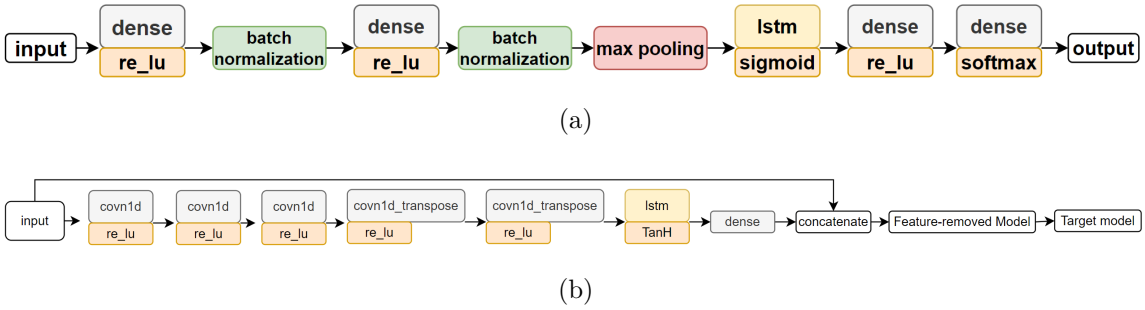


Figure 4.2: The target model architecture and the architecture of the generator AE model

In this study, we assume that the attacker has compromised one of the three devices and can change the values of all the sensors on the compromised device. We assumed that the victim did not know the compromised sensor, but was aware that the sensors in the same device carry the same risk.

We generate the attack by using the attack generator trained for the target model (FRM) [48]. Figure 4.2b shows the structure of the generator. The generator is trained so that the outputs of the FRM using all features become the class the attacker wants. When training the generator, we assume that the attacker has enough information about the target model (FRM), and we use the data from subjects 1 to 8. For training the generator, we used the Adam optimizer with a learning rate of 0.002 and a batch size of 256 for 25 epochs.

On top of TensorFlow and Keras, we constructed the FRM, original model, and generator models and trained them with four NVIDIA Quadro RTX 6000 GPU cards.

4.5.2 Property of the Feature Removable Model Without Attack

Before demonstrating our method to detect the attacks, we present the properties of the FRM, comparing them with the original model. We define the identified class as the class whose corresponding output exceeds the threshold, where the threshold = 0.60. We use *Precision* and *Recall* as metrics for this comparison. The definitions of *Precision* and *Recall*

Subject	**Standing**	**Lying Down**	**Walking**	**Climbing Stairs**	**Waist Bends Forward**	**Frontal Elevation of Arms**	**Knees Bending**	**Cycling**	**Jogging**	**Running**	**Jump Front and Back**
1	3072	3072	3072	3072	3072	3132	3278	3179	3072	3072	1024
2	3072	3072	3072	3072	3132	3132	3380	3430	3072	3072	1024
3	3072	3072	3072	3072	3132	3132	3266	3379	3175	3072	1024
4	3072	3072	3072	3072	3132	3132	3266	3379	3175	3072	1024
5	3072	3072	3072	3072	3132	3132	7265	7141	2784	3072	1024
6	3072	3072	3072	3072	2766	2894	2816	3072	3072	3072	1024
7	3072	3072	3072	3072	2766	2894	2816	3072	3072	3072	1024
8	3072	3072	3072	3072	2458	2765	2867	3072	3072	3072	1025
9	3072	3072	3072	3072	2458	2765	2867	3072	3072	3072	1025
10	3072	3072	3072	3072	2458	2765	2867	3072	3072	3072	1025

Table 4.2: Data that we used for each class and individual.

Activities	**Original Model** *Precision*	*Recall*	**Features-removed Model** *Precision*	*Recall*
Standing	1.00	1.00	0.98	1.00
Sitting	1.00	1.00	0.99	1.00
Lying down	1.00	1.00	0.99	1.00
Walking	1.00	1.00	0.97	1.00
Climbing stairs	0.98	1.00	0.96	0.99
Waist bends forward	1.00	1.00	0.99	1.00
Frontal elevation of arms	0.98	1.00	0.99	1.00
Knees bending	1.00	0.96	0.99	0.97
Cycling	0.96	1.00	0.98	1.00
Jogging	0.98	0.93	0.95	1.00
Running	0.94	0.97	0.98	1.00
Jump front and back	0.95	0.97	0.98	1.00
Average	**0.98**	**0.98**	**0.98**	**0.99**

Table 4.3: The full results of the original model compared with the features-removed model.

are:

$$Precision = \frac{tp}{tp + fp} \tag{4.3}$$

$$Recall = \frac{tp}{tp + fn} \tag{4.4}$$

where tp is the number of activities that are correctly identified as the target class, fp is the number of activities that are identified as the target class but whose actual class is different, and fn is the number of activities whose actual class is the target class but that are not identified as the target class. The FRM can output probabilities larger than the threshold for multiple classes. If probabilities for multiple classes are lager than the threshold, we independently count all classes to calculate tp and fp.

Table 4.3 shows the comparison between the original model and FRM using all features. Table 4.3 indicates that the FRM is similar to the original model, while the *Precision* of the FRM is slightly lower than that of the original model. The *Precision* of the FRM is caused by that the FRM allows large probabilities for multiple classes. However, the FRM achieves a sufficiently high *Precision*, which is greater than 0.95.

4.5.3 Property of the Attack

In this subsection, we evaluate the attacks on FRM using all the features. Figure 4.3 shows the attack success rates based on sensors located on the ankle, wrist, and chest, respectively.

In this Figure 4.3, we consider attacks that result in the outputted probability of the FRM using all features for the target class exceeding 0.60 as successful attacks. We define the attack success ratio as $\frac{N_{\text{success}}}{N_{\text{attack}}}$, where N_{success} is the number of successful attacks and N_{attack} is the number of generated attacks.

Figure 4.3 shows that the ankle, wrist, and chest sensors group is susceptible to a wide range of adversarial attacks. Many of the ground-truth classes were highly vulnerable in this figure. However, Figure 4.3 also shows that some attacks were difficult to succeed. The rest of this section focuses on the successful attacks.

4.5.4 Accuracy of Detection

In this subsection 4.5.4, we evaluate our attack detection by using two metrics, false negative rate (FNR) and false positive rate (FPR) defined by

$$\text{FNR} = \frac{\text{FN}}{\text{FN} + \text{TP}} \tag{4.5}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{4.6}$$

where TP is the number of sensor-based AEs that are correctly detected by our method, FN is the number of sensor-based AEs that cannot be detected, FP is the number of clean data that are mistakenly detected as sensor-based AEs, and TN is the number of clean data that are not detected by our method.

Our detection method has two parameters: α and β. These parameters have a large impact on the detection of sensor-based AEs. In this evaluation, we changed these parameters to investigate their impact on both FNR and FPR. Figure 4.4 shows the results. This figure shows a trade-off between the FPR and FNR. A higher value of these parameters tends to decrease FPR at the expense of an increased FNR.

Sensors attach	Ground Truth Class	Target Attack Class											
		St	*Si*	*Ly*	*Wa*	*Cs*	*Wb*	*Fe*	*Kb*	*Cy*	*Jo*	*Ru*	*Ju*
Ankle	Standing (*St*)		1.00	0.00	1.00	1.00	0.00	1.00	1.00	0.90	0.80	0.00	1.00
	Sitting (*Si*)	0.20		1.00	0.30	0.90	1.00	1.00	1.00	0.83	0.80	1.00	1.00
	Lying down (*Ly*)	1.00	1.00		0.00	1.00	1.00	1.00	1.00	1.00	0.40	0.00	1.00
	Walking (*Wa*)	0.00	1.00	0.00		0.80	1.00	1.00	1.00	0.00	0.00	1.00	0.00
	Climbing stairs (*Cs*)	0.50	0.90	1.00	1.00		1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Waist bends forward (*Wb*)	0.70	1.00	1.00	1.00	1.00		1.00	1.00	1.00	0.00	1.00	1.00
	Frontal elevation of arms (*Fe*)	0.00	1.00	1.00	1.00	0.80	1.00		0.00	1.00	1.00	1.00	1.00
	Knees bending (*Kb*)	1.00	1.00	1.00	1.00	1.00	0.90	1.00		0.90	0.90	1.00	1.00
	Cycling (*Cy*)	0.00	1.00	1.00	1.00	1.00	0.00	1.00	0.80		1.00	1.00	1.00
	Jogging (*Jo*)	1.00	1.00	0.70	0.00	1.00	1.00	1.00	1.00	1.00		1.00	1.00
	Running (*Ru*)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00		1.00
	Jump front and back (*Ju*)	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	
Wrist	Standing (*St*)		1.00	0.00	0.00	1.00	0.00	1.00	1.00	0.00	0.20	0.00	0.00
	Sitting (*Si*)	1.00		0.00	0.00	1.00	0.60	0.50	1.00	0.80	0.00	0.00	0.10
	Lying down (*Ly*)	0.00	1.00		1.00	0.50	0.00	0.00	0.70	1.00	0.00	0.00	1.00
	Walking (*Wa*)	0.60	1.00	0.60		1.00	0.00	1.00	0.30	0.00	0.00	1.00	0.00
	Climbing stairs (*Cs*)	1.00	0.30	0.00	0.00		0.90	0.00	0.00	1.00	1.00	1.00	0.00
	Waist bends forward (*Wb*)	1.00	0.70	0.40	0.30	0.40		0.60	1.00	1.00	0.70	1.00	1.00
	Frontal elevation of arms (*Fe*)	1.00	0.70	1.00	1.00	0.90	0.00		0.20	0.00	0.90	0.90	1.00
	Knees bending (*Kb*)	0.00	1.00	1.00	0.00	0.70	1.00	1.00		1.00	1.00	1.00	1.00
	Cycling (*Cy*)	0.00	1.00	1.00	0.00	1.00	1.00	0.00	1.00		1.00	1.00	1.00
	Jogging (*Jo*)	0.10	0.00	0.00	0.70	0.00	0.20	0.00	0.00	1.00		1.00	1.00
	Running (*Ru*)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.80		1.00
	Jump front and back (*Ju*)	1.00	1.00	0.30	0.00	0.60	0.00	0.00	0.00	1.00	1.00	1.00	
Chest	Standing (StSt)		0.80	0.00	1.00	0.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00
	Sitting (*Si*)	1.00		1.00	0.40	0.80	0.00	0.80	0.00	0.50	0.00	0.00	0.00
	Lying down (LyLy)	0.80	0.00		1.00	0.70	1.00	1.00	0.10	0.40	1.00	0.00	0.10
	Walking (*Wa*)	0.00	1.00	0.00		0.00	1.00	0.00	0.00	0.00	0.00	1.00	0.00
	Climbing stairs (*Cs*)	0.50	0.90	1.00	1.00		0.20	0.70	0.00	1.00	1.00	0.00	1.00
	Waist bends forward (*Wb*)	0.00	0.00	1.00	1.00	1.00		1.00	0.00	0.00	0.00	0.00	0.00
	Frontal elevation of arms (*Fe*)	0.00	0.00	0.00	0.00	0.00	1.00		0.00	1.00	0.00	1.00	0.00
	Knees bending (*Kb*)	0.60	0.00	1.00	1.00	1.00	0.00	1.00		1.00	0.00	0.00	0.70
	Cycling (*Cy*)	1.00	0.80	1.00	0.90	0.70	1.00	1.00	0.60		0.00	0.70	0.00
	Jogging (JoJo)	0.00	0.00	0.00	0.70	0.40	1.00	0.00	0.00	0.10		1.00	1.00
	Running (*Ru*)	1.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	0.00	0.00		0.50
	Jump front and back (*Ju*)	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	0.60	0.00	1.00	

Figure 4.3: The success ratio of generated sensor-based AEs

This figure also shows that we can achieve the FNR less than 0.03 without letting the FPR exceed 0.06.

4.5.5 Accuracy of Detection of Sensors Used in Sensor-based AEs.

Our method also detects the sensors used in sensor-based AEs. In this subsection 4.5.5, we evaluate the accuracy of the detection by using two metrics, *Precision Of Detected Sensors* and *Recall Of Detected Sensors* as defined by

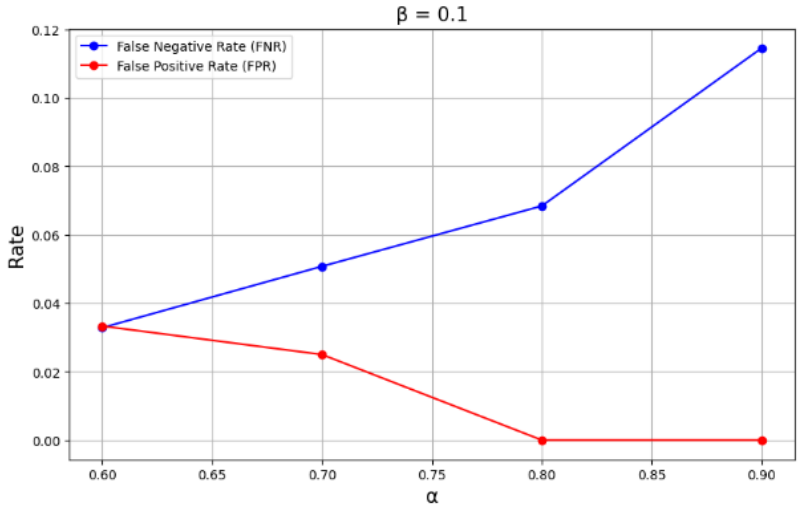
$$Precision\ Of\ Detected\ Sensors = \frac{Truly\ Detected\ Sensors}{Truly\ Detected\ Sensors + Falsely\ Detected\ Sensors} \tag{4.7}$$

$$Recall\ Of\ Detected\ Sensors = \frac{Truly\ Detected\ Sensors}{Truly\ Detected\ Sensors + Misses\ Detected\ Sensors} \tag{4.8}$$

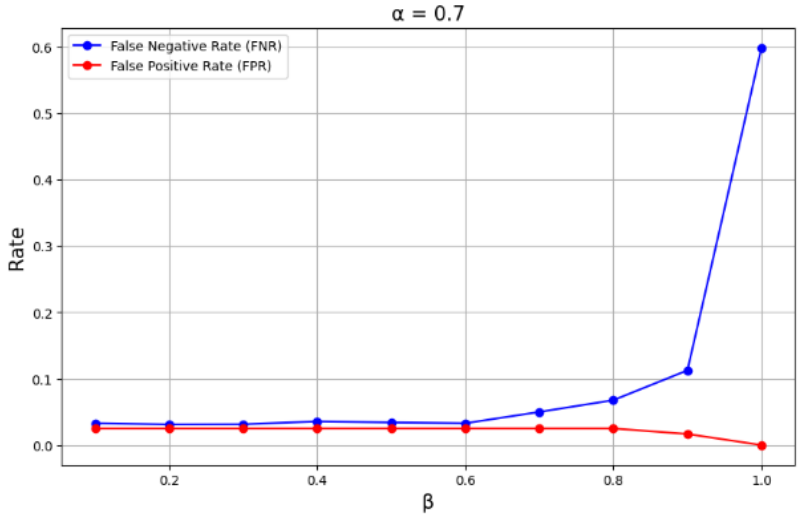
where *Truly Detected Sensors* is the number of sensors that are correctly detected as the sensors used in the sensor-based AEs, *Falsely Detected Sensors* is the number of sensors detected as the sensors used in the sensor-based AEs but are not used in the AEs, and *MissedDetected* is the number of sensors that are used in the sensor-based AEs but are not detected as the sensors used in the AEs. If our method detects all sensors used in the AEs, the *Recall Of Detected Sensors* becomes 1.00. It is important to detect sensors used by attackers as suspicious, even if some legitimate sensors are mistakenly detected. Therefore, the *Recall Of Detected Sensors* is more important than *Precision Of Detected Sensors*.

Figure 4.5 shows the *Recall Of Detected Sensors*. In this Figure, the green cells are the cells whose values are larger than 0.50, the yellow cells are the cells whose values are larger than 0.50 and less than 0.80, and the red cells are the cells whose values are larger than 0.80.

Figure 4.5 shows that our method achieves high *Recall Of Detected Sensors*. *Recall Of Detected Sensors* calculated for all cases is 0.92. That is, most of the sensors used in the AEs can be detected by our method. This is because our method uses the features of the attack, which causes the prediction results to change when the attacker uses the features from sensors. This feature cannot be avoided by the attacker because they are unable to



(a)



(b)

Figure 4.4: The results of impact when FNR and FPR were α value is changing AND $\beta = 0.1$: and result of impact when the β value is changing and $\alpha = 0.7$

Sensors attach	**Ground Truth Class**	**Target Attack Class**											
		St	*Si*	*Ly*	*Wa*	*Cs*	*Wb*	*Fe*	*Kb*	*Cy*	*Jo*	*Ru*	*Ju*
Ankle	Standing (*St*)		1.00	N/A	1.00	1.00	N/A	1.00	1.00	1.00	1.00	N/A	1.00
	Sitting (*Si*)	1.00		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Lying down (*Ly*)	1.00	1.00		N/A	1.00	1.00	1.00	1.00	1.00	1.00	N/A	1.00
	Walking (*Wa*)	N/A	1.00	N/A		1.00	1.00	1.00	1.00	N/A	N/A	1.00	N/A
	Climbing stairs (*Cs*)	1.00	1.00	1.00	1.00		1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Waist bends forward (*Wb*)	1.00	1.00	1.00	1.00	1.00		1.00	1.00	1.00	N/A	1.00	1.00
	Frontal elevation of arms (*Fe*)	N/A	1.00	1.00	1.00	1.00	1.00		N/A	1.00	1.00	1.00	1.00
	Knees bending (*Kb*)	1.00	1.00	1.00	1.00	1.00	0.91	1.00		0.91	1.00	1.00	1.00
	Cycling (*Cy*)	N/A	1.00	1.00	1.00	1.00	N/A	1.00	1.00		1.00	1.00	1.00
	Jogging (*Jo*)	1.00	1.00	1.00	1.00	0.96	1.00	1.00	1.00	1.00		1.00	1.00
	Running (*Ru*)	1.00	1.00	N/A	N/A	1.00	1.00	1.00	1.00	1.00	1.00		1.00
	Jump front and back (*Ju*)	N/A	1.00	N/A	N/A	1.00	1.00	1.00	1.00	1.00	1.00	1.00	
Wrist	Standing (*St*)		0.79	N/A	N/A	1.00	N/A	1.00	1.00	N/A	1.00	N/A	N/A
	Sitting (*Si*)	1.00		N/A	N/A	1.00	1.00	1.00	1.00	1.00	N/A	N/A	1.00
	Lying down (*Ly*)	N/A	1.00		1.00	1.00	N/A	N/A	1.00	1.00	N/A	N/A	1.00
	Walking (*Wa*)	1.00	1.00	1.00		1.00	N/A	1.00	1.00	N/A	N/A	1.00	N/A
	Climbing stairs (*Cs*)	1.00	1.00	N/A	N/A		1.00	N/A	N/A	1.00	1.00	1.00	N/A
	Waist bends forward (*Wb*)	1.00	1.00	1.00	1.00	1.00		1.00	1.00	1.00	1.00	1.00	1.00
	Frontal elevation of arms (*Fe*)	1.00	0.95	1.00	1.00	1.00	N/A		1.00	N/A	1.00	1.00	1.00
	Knees bending (*Kb*)	N/A	1.00	1.00	N/A	1.00	1.00	1.00		1.00	1.00	1.00	1.00
	Cycling (*Cy*)	N/A	1.00	1.00	N/A	1.00	1.00	N/A	1.00		1.00	1.00	1.00
	Jogging (*Jo*)	1.00	N/A	N/A	1.00	N/A	1.00	N/A	N/A	1.00		0.82	1.00
	Running (*Ru*)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1.00	0.98		1.00
	Jump front and back (*Ju*)	1.00	1.00	1.00	N/A	1.00	N/A	N/A	N/A	1.00	1.00	1.00	
Chest	Standing (*St*)		1.00	N/A	1.00	N/A	1.00	1.00	N/A	N/A	N/A	N/A	N/A
	Sitting (*Si*)	1.00		1.00	1.00	1.00	N/A	1.00	N/A	1.00	N/A	N/A	N/A
	Lying down (*Ly*)	1.00	1.00		1.00	1.00	1.00	1.00	1.00	1.00	1.00	N/A	1.00
	Walking (*Wa*)	N/A	1.00	N/A		N/A	1.00	N/A	N/A	N/A	N/A	1.00	N/A
	Climbing stairs (*Cs*)	1.00	0.86	1.00	1.00		1.00	1.00	N/A	1.00	1.00	N/A	1.00
	Waist bends forward (*Wb*)	N/A	N/A	1.00	1.00	1.00		1.00	N/A	N/A	N/A	N/A	N/A
	Frontal elevation of arms (*Fe*)	N/A	N/A	N/A	N/A	N/A	1.00		N/A	1.00	N/A	1.00	N/A
	Knees bending (*Kb*)	1.00	N/A	1.00	1.00	1.00	N/A	1.00		1.00	N/A	N/A	1.00
	Cycling (*Cy*)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00		N/A	1.00	N/A
	Jogging (*Jo*)	N/A	N/A	N/A	1.00	1.00	1.00	N/A	N/A	1.00		1.00	1.00
	Running (*Ru*)	1.00	N/A	N/A	N/A	N/A	1.00	1.00	N/A	N/A	N/A		1.00
	Jump front and back (*Ju*)	N/A	N/A	N/A	N/A	N/A	1.00	1.00	N/A	1.00	N/A	1.00	

Figure 4.5: Recall Of Detected Sensors

alter the results of the FRM, which extracts the features from the sensors used by the attacker.

However, the *Recall Of Detected Sensors* for some cases are not 1.00; for example, it is 0.79 in the case of the AEs from standing class to sitting class by compromising the wrist sensor. Such misdetection occurs when the target class cannot be distinguished from the actual class. In this case, the FRM, excluding the values from the sensors used in the AEs, outputs a high probability even for the target class. One approach to solving this problem is to add more sensors to make the system more redundant so that any class can be distinguished even if we exclude some sensors, which are elaborated in detail in Sections 4.6 and 4.7 of this chapter.

Figure 4.6 shows *Precision Of Detected Sensors*. This figure indicates that our method achieves high *Precision Of Detected Sensors* in most cases. The *Precision Of Detected Sensors* calculated for all cases is 0.72.

But *Precision Of Detected Sensors* becomes low in some cases. For example, the *Precision Of Detected Sensors* for the case that values from the chest sensor device are changed so that the state of lying down is miss-classified into the state of knee bending is 0.22. In such cases, inconsistencies are found by extracting the features from the other sensors. However, even in such cases, we can successfully detect sensors used in the AEs as suspicious sensors. Therefore, our method can be used as a trigger to check suspicious sensors.

4.5.6 Mitigation of Sensor-based AEs by Excluding the Detected Sensors

The FRM can be used after detection of sensor-based AEs and compromised sensors, because the FRM can output the classification results even if some features are excluded. Therefore, we demonstrate the performance of the FRM by excluding the values from the detected sensors.

Table 4.4 presents the *Precision* (P) and *Recall* (R) of the FRM excluding the values from the sensors used in the AEs. This table also includes the results of the original model in the cases without attacks.

Sensors attach	**Ground Truth Class**	**Target Attack Class**											
		St	*Si*	*Ly*	*Wa*	*Cs*	*Wb*	*Fe*	*Kb*	*Cy*	*Jo*	*Ru*	*Ju*
Ankle	Standing (*St*)		1.00	N/A	0.41	0.39	N/A	0.64	1.00	1.00	0.67	N/A	1.00
	Sitting (*Si*)	0.64		1.00	0.45	0.60	0.90	1.00	1.00	0.66	0.49	1.00	1.00
	Lying down (*Ly*)	1.00	1.00		N/A	0.82	1.00	1.00	1.00	1.00	0.39	N/A	1.00
	Walking (*Wa*)	N/A	1.00	N/A		1.00	1.00	1.00	1.00	N/A	N/A	1.00	N/A
	Climbing stairs (*Cs*)	1.00	1.00	1.00	0.21		1.00	1.00	1.00	1.00	0.54	1.00	1.00
	Waist bends forward (*Wb*)	1.00	1.00	1.00	0.50	0.50		1.00	1.00	1.00	N/A	1.00	1.00
	Frontal elevation of arms (*Fe*)	N/A	1.00	1.00	1.00	0.53	1.00		N/A	1.00	0.61	1.00	1.00
	Knees bending (*Kb*)	0.91	1.00	1.00	0.91	0.83	1.00	1.00		1.00	1.00	1.00	1.00
	Cycling (*Cy*)	N/A	1.00	1.00	0.57	0.83	N/A	1.00	1.00		0.42	1.00	1.00
	Jogging (*Jo*)	1.00	1.00	1.00	1.00	0.66	1.00	1.00	1.00	1.00		1.00	1.00
	Running ()	1.00	1.00	N/A	N/A	0.78	1.00	1.00	1.00	1.00	0.57		1.00
	Jump front and back (*Ju*)	N/A	1.00	N/A	N/A	0.66	1.00	1.00	1.00	1.00	1.00	1.00	
Wrist	Standing (*St*)		1.00	N/A	N/A	0.60	N/A	1.00	0.68	N/A	0.64	N/A	N/A
	Sitting (*Si*)	1.00		N/A	N/A	0.75	0.57	0.50	0.90	0.84	N/A	N/A	1.00
	Lying down (*Ly*)	N/A	1.00		0.64	0.90	N/A	N/A	1.00	1.00	N/A	N/A	0.60
	Walking (*Wa*)	1.00	1.00	1.00		1.00	N/A	1.00	0.39	N/A	N/A	1.00	N/A
	Climbing stairs (*Cs*)	1.00	1.00	N/A	N/A		1.00	N/A	N/A	1.00	0.87	1.00	N/A
	Waist bends forward (*Wb*)	1.00	1.00	1.00	0.59	0.57		0.67	0.72	1.00	0.48	1.00	1.00
	Frontal elevation of arms (*Fe*)	1.00	1.00	0.80	0.68	0.55	N/A		0.44	N/A	0.70	1.00	1.00
	Knees bending (*Kb*)	N/A	1.00	1.00	N/A	1.00	1.00	1.00		1.00	1.00	1.00	1.00
	Cycling (*Cy*)	N/A	1.00	1.00	N/A	1.00	1.00	N/A	1.00		0.57	1.00	1.00
	Jogging (JoJo)	1.00	N/A	N/A	0.58	N/A	1.00	N/A	N/A	1.00		1.00	1.00
	Running (*Ru*)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1.00	0.54		1.00
	Jump front and back (*Ju*)	1.00	1.00	1.00	N/A	0.56	N/A	N/A	N/A	1.00	1.00	1.00	
Chets	Standing (StSt)		0.69	N/A	0.36	N/A	1.00	0.56	N/A	N/A	N/A	N/A	N/A
	Sitting (*Si*)	0.36		1.00	0.36	0.69	N/A	0.33	N/A	0.53	N/A	N/A	N/A
	Lying down (*Ly*)	0.33	0.22		0.48	0.44	1.00	1.00	0.22	1.00	0.64	N/A	1.00
	Walking (*Wa*)	N/A	1.00	N/A		N/A	1.00	N/A	N/A	N/A	N/A	1.00	N/A
	Climbing stairs (*Cs*)	0.55	1.00	1.00	0.60		1.00	0.63	N/A	1.00	0.82	N/A	1.00
	Waist bends forward (*Wb*)	N/A	N/A	0.83	0.50	1.00		0.48	N/A	N/A	N/A	N/A	N/A
	Frontal elevation of arms (*Fe*)	N/A	N/A	N/A	N/A	N/A	1.00		N/A	0.56	N/A	1.00	N/A
	Knees bending (*Kb*)	0.55	N/A	1.00	1.00	0.85	N/A	1.00		1.00	N/A	N/A	1.00
	Cycling (*Cy*)	1.00	1.00	1.00	0.71	0.47	1.00	1.00	0.48		N/A	1.00	N/A
	Jogging (*Jo*)	N/A	N/A	N/A	0.44	0.24	1.00	N/A	N/A	0.36		1.00	0.85
	Running (*Ru*)	1.00	N/A	N/A	N/A	N/A	1.00	1.00	N/A	N/A	N/A		0.58
	Jump front and back (*Ju*)	N/A	N/A	N/A	N/A	N/A	1.00	1.00	N/A	0.60	N/A	1.00	

Figure 4.6: Precision Of Detected Sensors

Ground Truth Class	Original Model		Ankle		Wrist		Chest	
	P	R	P	R	P	R	P	R
Standing	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.99
Sitting	1.00	1.00	0.99	0.99	0.76	0.99	1.00	1.00
Lying down	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99
Walking	1.00	1.00	1.00	0.99	1.00	1.00	1.00	0.99
Climbing stairs	0.99	1.00	0.71	0.99	1.00	1.00	0.99	0.99
Waist bends forward	1.00	1.00	1.00	1.00	0.99	0.99	1.00	1.00
Frontal elevation of arms	0.98	1.00	0.99	0.99	0.99	0.99	0.99	0.99
Knees bending	1.00	0.96	1.00	0.99	0.99	0.99	0.75	0.99
Cycling	0.96	1.00	1.00	0.99	0.99	0.99	0.99	0.99
Jogging	0.98	0.93	1.00	0.99	0.99	0.99	0.99	0.99
Running	0.94	0.97	0.99	0.99	1.00	1.00	0.99	0.99
Jump front and back	0.95	0.97	0.99	0.99	0.99	0.99	0.99	0.99
Average	**0.98**	**0.98**	**0.97**	**0.99**	**0.97**	**0.99**	**0.99**	**0.99**

Table 4.4: Performance comparison of the original target model without AEs and FRM exluding the values from the sensors used in the AEs

Table 4.4 indicates that the FRM excluding sensors used in the AEs achieves high *Precision* and *Recall* in most cases. In some cases, the *Precision* becomes low. For example, *Precision* for the class of climbing stairs is 0.71 when excluding the values from the sensors of the ankle device. This is because the values from the ankle device are essential to distinguish the classes. Even in this case, the FRM can output the actual classes by outputting large probabilities for multiple possible classes.

4.6 Criticality of Sensors.

In the previous sections 4.5, we proposed a method to detect the sensors used by the attacks. However, reactive defense method based on detection has limitations. If some critical sensors that are necessary to distinguish required states are compromised by the attacker, we cannot obtain the suitable output. On the other hand, a system with enough redundancy can work after removing the features from the sensors used in the AEs. That is, we need a metric to check if the system has enough redundancy.

In this section 4.6, we define groups of sensors that might be compromised by the same attacker, and we propose a metric called *criticality* that indicates how important each group of sensors are for classification between two classes. Based on the criticality, we can make the system robust against sensor-based AEs by interactively adding sensors so as to decrease the criticality of any groups of sensors for the classes that must be distinguished.

4.6.1 Definition of Criticality

In this subsection 4.6.1, we define groups of sensors that might be compromised by the same attacker and define a metric called *criticality* that indicates how important each group of sensors is for classification between two classes. We define the criticality based of if the classes can be distinguished without the sensors.

The FRM is also useful to check if the class can be distinguished without the sensors, because the FRM allows us to select the features used as an input into the model. Therefore, we define the criticality based on the output of the FRM. We can check if the group of sensors g is critical to distinguish the classes i and j by the output of the FRM without using the values from the sensors in the group g. If the sensors in the group g is necessary to distinguish the class i from j, the output probability of the FRM without using the values from the group of the sensor g for the class j becomes large when the data whose actual class is i.

So we define the criticality of the sensor group g for the classes i and j by

$$C_s(i,j) = \frac{\sum_{d \in D_i} Y_j^g(d)}{|D_i|} \tag{4.9}$$

where D_i is the set of data whose actual class is i and $Y_j^g(d)$ is the output probability of the FRM without using the values from the sensor group g for the class j.

A large value of $C_s(i,j)$ indicates that it is difficult to distinguish the class i from the class j without values from the sensor s.

4.6.2 Example of Criticality

We train the FRM by using the Adam optimizer with a learning rate of 0.001 and batches of 32 for 100 epochs. We set weights in Eq 4.2 so that w(0) is 1.0 and w(1) is 20.0. We use the MHealth dataset [91], which includes 12 distinct physical activities for ten individuals.

In this subsection 4.6.2, we evaluate the criticality of the system used in the previous section. This system recognize human activity from three devices mounted at the chest, left ankle, and right wrist. We consider the risk that each of the sensor devices can be compromised by an attacker. The values from the compromised sensors can be changed by the attacker. So, the system should be able to distinguish the classes without using one of the sensor devices.

Figure 4.7 shows the criticality calculated by considering these risks. In this figure, we colored red for the cell with a criticality higher than 0.9, and yellow for the cell with a criticality higher than 0.5. This figure indicates that the criticality of most class pairs was very low. That is, this system have enough redundancy and can distinguish such classes without using one of the sensor devices.

However, ”Walking” and ”Climbing stairs” are difficult to be distinguished without the ankle sensor device. ”Sitting”, ”Frontal elevation of arms” and ”Standing” are also difficult to be distinguished without the wrist sensor device. That is, if these classes are required to be distinguished, we need to add more sensors to make this system robust against sensor-based AEs.

4.7 Discussion Toward Robust System Against Sensor-based AEs.

The system is robust against sensor-based AEs, if the criticality of any risk groups is small for all class pairs required to be distinguished. However, it may requires a large cost to achieve that any classes can be distinguished in any cases of the risks. Therefore, we should focus on the risks with high probability and the important class pairs.

	Ground Truth Class	*St*	*Si*	*Ly*	*Wa*	*Cl*	*WB*	*FE*	*KB*	*Cy*	*Jo*	*Ru*	*JF*
	Standing (*St*)	**N/A**	0.14	0.00	0.00	0.00	0.00	0.76	0.00	0.00	0.00	0.00	0.00
	Sitting (*Si*)	0.03	**N/A**	0.00	0.00	0.00	0.00	0.08	0.00	0.00	0.00	0.00	0.00
	Lying down (*Ly*)	0.00	0.00	**N/A**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Walking (*Wa*)	0.00	0.00	0.00	**N/A**	0.91	0.00	0.00	0.59	0.00	0.00	0.02	0.00
	Climbing stairs (*Cl*)	0.00	0.00	0.00	0.21	**N/A**	0.01	0.00	0.1	0.32	0.00	0.02	0.00
Ankle	Waist bends forward (*WB*)	0.00	0.00	0.00	0.00	0.03	**N/A**	0.00	0.63	0.00	0.00	0.00	0.00
	Frontal elevation of arms (*FE*)	0.01	0.00	0.00	0.00	0.00	0.00	**N/A**	0.00	0.00	0.00	0.00	0.00
	Knees bending (*KB*)	0.00	0.00	0.00	0.02	0.04	0.57	0.00	**N/A**	0.00	0.00	0.00	0.00
	Cycling (*Cy*)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**N/A**	0.00	0.00	0.00
	Jogging (*Jo*)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**N/A**	0.44	0.03
	Running (*Ru*)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.29	**N/A**	0.00
	Jump front and back (*JF*)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.13	0.48	**N/A**
	Standing (*St*)	**N/A**	0.05	0.00	0.00	0.00	0.00	0.24	0.00	0.00	0.00	0.00	0.00
	Sitting (*Si*)	0.97	**N/A**	0.00	0.00	0.00	0.00	0.92	0.00	0.00	0.00	0.00	0.00
	Lying down (*Ly*)	0.00	0.00	**N/A**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Walking (*Wa*)	0.00	0.00	0.00	**N/A**	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Climbing stairs (*Cl*)	0.00	0.00	0.00	0.00	**N/A**	0.00	0.00	0.06	0.12	0.01	0.00	0.00
Wrist	Waist bends forward (*WB*))	0.00	0.00	0.00	0.00	0.00	**N/A**	0.05	0.01	0.06	0.00	0.00	0.00
	Frontal elevation of arms (*FE*)	0.99	0.49	0.00	0.00	0.00	0.00	**N/A**	0.00	0.00	0.00	0.00	0.00
	Knees bending (*KB*)	0.01	0.01	0.00	0.00	0.00	0.04	0.01	**N/A**	0.17	0.00	0.00	0.00
	Cycling (*Cy*)	0.00	0.00	0.00	0.00	0.15	0.00	0.00	0.00	**N/A**	0.00	0.00	0.00
	Jogging (*Jo*)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**N/A**	0.31	0.04
	Running (*Ru*)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.16	**N/A**	0.00
	Jump front and back (*JF*)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	**N/A**
	Standing (*St*)	**N/A**	0.14	0.00	0.00	0.00	0.00	0.76	0.00	0.00	0.00	0.00	0.00
	Sitting (*Si*)	0.03	**N/A**	0.00	0.00	0.00	0.00	0.08	0.00	0.00	0.00	0.00	0.00
	Lying down (*Ly*)	0.00	0.00	**N/A**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Walking (*Wa*)	0.00	0.00	0.00	**N/A**	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Climbing stairs (*Cl*)	0.00	0.00	0.00	0.00	**N/A**	0.00	0.01	0.00	0.00	0.02	0.00	0.00
Chest	Waist bends forward (*WB*)	0.00	0.00	0.00	0.00	0.03	**N/A**	0.00	0.04	0.00	0.00	0.00	0.00
	Frontal elevation of arms (*FE*)	0.00	0.00	0.00	0.00	0.00	0.00	**N/A**	0.00	0.00	0.00	0.00	0.00
	Knees bending (*KB*)	0.00	0.00	0.00	0.00	0.00	0.23	0.00	**N/A**	0.00	0.00	0.00	0.00
	Cycling (*Cy*)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	**N/A**	0.01	0.06	0.00
	Jogging (*Jo*)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**N/A**	0.08	0.02
	Running (*Ru*)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	**N/A**	0.00
	Jump front and back (*JF*)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.43	**N/A**

Figure 4.7: The criticality of each sensor device for each class pair

Considering the above points, we can create a robust system against sensor-based AEs as follows.

4.7.1 Building a System

We make a system based on the existing sensors. Then, we train the FRM by using the training data from the existing sensors.

4.7.2 Assessment of Importance of Class Identification

We assess the importance of the class identification. In some applications, misclassification of some similar classes does not have a significant impact. Considering that, we need to evaluate the importance of the distinguishment of classes and focus on the important class pairs.

4.7.3 Assessment of Risk

We also assess the possible risk of compromised sensors. The sensors with the same location, the same kind of sensors, or the sensor devices with the same OS might be compromised by the same attacker. We consider the cases that such sensors are compromised. We asses risk of each case. We also define the sensors compromised in each case.

4.7.4 Evaluation Based on Criticality and Update of the System

We then calculate the criticality for the set of the sensors whose risk to be compromised is high or the important class pairs. If the calculated criticality exceeds the threshold, we regard the current system as the system vulnerable to the sensor-based AEs and add more sensors. After adding the sensors, we asses the risk of compromised sensors and evaluate the system again. By continuing the addition of the sensors, we make the system robust against the sensor-based AEs.

4.8 Conclusions.

In this thesis, we propose a method for detecting the sensors used in sensor-based AEs. Our method utilizes the features of sensor-based AEs that attackers cannot avoid. The output of the ML model changed when the values from the sensors used in the AE were incorporated. To assess the impact of sensor values, we introduced a feature-removable model (FRM), allowing feature selection for use. The FRM outputs the possible classes that are classified using the selected features. By comparing the FRM results with different feature selections, we can detect the inconsistencies and identify the sensors that cause them. After the sensors used in the AEs are detected, they can be verified and replaced. Furthermore, FRM can be employed post-detection, excluding features from detected sensors to mitigate attack impacts, although this may reduce the accuracy.

Through an experimental evaluation, a model for human activity recognition was tested using three devices attached to the user's chest, wrist, and ankle. Our method successfully detected the sensors used in AEs, even though one-third of the sensors were compromised by an attacker.

In multisensor systems, the risk of vulnerable sensor devices producing AEs increases with the number of sensors. Protecting all sensor devices is challenging, necessitating methods to safeguard ML models even when some sensors are compromised. One approach is to detect and remove the sensors involved in the attacks. However, this reactive defense system has limitations, particularly when the critical sensors necessary for accurate state distinction are compromised.

Furthermore, we discuss a proactive strategy to enhance system robustness against AEs. A system with sufficient redundancy can function effectively after removing features from the sensors involved in the AEs. Hence, we introduced a metric for assessing system redundancy. We defined sensor groups potentially compromised by the same attacker and proposed a metric called 'criticality' to gauge the importance of each sensor group in class classification. By interactively adding sensors to reduce the criticality of any group for essential class distinctions, we can bolster system resilience against sensor-based AEs.

In future research, we will investigate the design of robust multisensor systems in real-world scenarios and demonstrate their robustness against sophisticated adversarial examples. This proactive approach ensures continued system robustness against potential adversarial challenges.

Chapter 5

Conclusion

As the information technologies become important infrastructures, cyberattacks have become major concerns. These attacks not only compromise information system integrity but can also lead to substantial economic losses, reputation damage, and physical safety risks. Therefore, the information systems should be more robust against such cyberattacks.

To make the system more robust against attacks, it is important to comprehend how existing attacks emerge and spread by analyzing them in detail by reconstructing events of cyberattacks [2, 6]. Based on the knowledge obtained by the analysis of the existing attacks, we can make the system more robust against cyberattacks.

In this thesis, we begin by analyzing ransomware cyberattacks using the lens of network forensics. We introduced a method to reconstruct the event chain of an attack using packet capture data, allowing us to identify infected hosts and their infection paths. We applied this method to the case of the CERBER ransomware. As a result, we found the event chain related to this ransomware; the user's search on bing.com led them to `www.homeimprovement.com`, a website compromised by cybercriminals. We also found that they used a pseudoDarkleech script to redirect visitors to a server that deployed the RIG Exploit Kit, resulting in the download of CERBER Ransomware.

We expand our discussion beyond traditional information systems, such as servers and personal computers, to include machine-learning models, which have increasingly become

targets for cyberattacks in recent years. The increase in the use of machine learning and sensor technologies necessitates enhanced protection for these models, ensuring robust defense against such attacks.

Adversarial examples (AEs) are one of the largest vulnerabilities of the machine learning models. In this attacks, an adversary generates inputs that causes a machine learning system to generate incorrect outputs. Especially in the system using multiple sensors, some sensors may be vulnerable and can be used to generate AEs. Even if an attacker can attack a machine learning model by using only a small part of sensors, the vulnerabilities of sensors have a significant risk. However, the impact of hacking a small part of the sensor has not been discussed thus far.

Therefore, we also discussed the impact of a small part of sensors on the machine learning models and demonstrate that the attacker can change the output of the ML models using multiple sensors if the attacker can manipulate the values from a part of sensors in this thesis. We call this attack *sensor-based AEs*. We performed experiments using the human activity recognition model with three sensor devices attached to the chest, wrist, and ankle of a user, and demonstrate that attacks are possible by hacking one of the sensor devices.

Then, we also discuss the countermeasure against sensor-based AEs. One of the approach to protecting the system from the sensor-based AEs is to protect all sensor devices. However, it is difficult to protect all sensor devices, because the risk of the existence of the vulnerable sensor devices increases as the number of sensor devices increases. Therefore, we need a method to protect machine learning models even if a part of sensors are compromised by the attacker.

Therefore, in this thesis, we proposed a new countermeasure focusing on the sensor-based AEs. This method detects sensor-based AEs and the sensors used by the attackers by checking the inconsistency of the output of the machine learning model obtained by changing the features used by the model. By detecting the sensors used by the attackers, we can check and replace them. Our method is based on the features of the sensor-based AEs that the attacker cannot avoid; the output of the machine learning model is altered when the values from the sensors used by the attacker are incorporated. We evaluated our

method using a human activity recognition model with sensors attached to the user's chest, wrist, and ankle. We demonstrate that our method can accurately detect sensors used by the attacker and achieves an average *Recall of Detection* of 0.92, and the average *Precision of Detection* is 0.72.

In our future work, we will explore the design of the robust system in real-world scenarios and demonstrate that the system is sufficiently robust against sophisticated attacks.

Bibliography

[1] A. Chowdhury, "Recent cyber security attacks and their mitigation approaches–an overview," in *Applications and Techniques in Information Security: 6th International Conference, ATIS 2016, Cairns, QLD, Australia, October 26-28, 2016, Proceedings 7*. Springer, 2016, pp. 54–65.

[2] R. Clarke and T. Youngstein, "Cyberattack on britain's national health service—a wake-up call for modern medicine," *New England Journal of Medicine*, vol. 377, no. 5, pp. 409–411, 2017.

[3] P. Shijo and A. Salim, "Integrated static and dynamic analysis for malware detection," *Procedia Computer Science*, vol. 46, pp. 804–811, 2015, proceedings of the International Conference on Information and Communication Technologies, ICICT 2014, 3-5 December 2014 at Bolgatty Palace Island Resort, Kochi, India. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050915002136

[4] E. Popoola and A. Adewumi, "Efficient feature selection technique for network intrusion detection system using discrete differential evolution and decision tree," *International Journal of Network Security*, vol. 19, no. 5, pp. 660–669, 2017.

[5] S. Das, Y. Liu, W. Zhang, and M. Chandramohan, "Semantics-based online malware detection: Towards efficient real-time protection against malware," *IEEE transactions on information forensics and security*, vol. 11, no. 2, pp. 289–302, 2015.

[6] H. Al-Mohannadi, Q. Mirza, A. Namanya, I. Awan, A. Cullen, and J. Disso, "Cyber-attack modeling analysis techniques: An overview," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, 2016, pp. 69–76.

[7] M. Groneberg, O. Poenicke, C. Mandal, and N. Treuheit, "Lidar and ai based surveillance of industrial process environments," *Transport and Telecommunication Journal*, vol. 24, no. 1, pp. 13–21, 2023.

[8] J. Havisto, T. Matselyukh, M. Paavola, S. Uusitalo, M. Savolainen, A. Sobrecueva González, A. Knobloch, and K. Bogdanov, "Golden ai data acquisition and processing platform for safe, sustainable and cost-efficient mining operations," in *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, 2021, pp. 5775–5778.

[9] M. Javaid, A. Haleem, I. H. Khan, and R. Suman, "Understanding the potential applications of artificial intelligence in agriculture sector," *Advanced Agrochem*, vol. 2, no. 1, pp. 15–30, 2023.

[10] S. Miao, Y. Dang, Q. Zhu, S. Li, M. Shorfuzzaman, and H. Lv, "A Novel Approach for Upper Limb Functionality Assessment Based on Deep Learning and Multimodal Sensing Data," *IEEE Access*, vol. 9, pp. 77 138–77 148, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9431090/

[11] E. Namazi, R. Mester, J. Li, C. Lu, M. Tang, and Y. Xiong, "Traffic Awareness Through Multiple Mobile Sensor Fusion," *IEEE Sensors Journal*, vol. 22, no. 12, pp. 11 903–11 914, june 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9764739/

[12] S. G. Finlayson, J. D. Bowers, J. Ito, J. L. Zittrain, A. L. Beam, and I. S. Kohane, "Adversarial attacks on medical machine learning," *Science*,

vol. 363, no. 6433, pp. 1287–1289, March 2019. [Online]. Available: https://www.sciencemag.org/lookup/doi/10.1126/science.aaw4399

[13] J. Classen, D. Wegemer, P. Patras, T. Spink, and M. Hollick, "Anatomy of a Vulnerable Fitness Tracking System," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 1, pp. 1–24, March 2018. [Online]. Available: https://dl.acm.org/doi/10.1145/3191737

[14] M. M. R. Monjur, J. Heacock, J. Calzadillas, M. S. Mahmud, J. Roth, K. Mankodiya, E. Sazonov, and Q. Yu, "Hardware Security in Sensor and its Networks," *Frontiers in Sensors*, vol. 3, May 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fsens.2022.850056/full

[15] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–11, 2015.

[16] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble Adversarial Training: Attacks and Defenses," *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pp. 1–20, May 2017. [Online]. Available: http://arxiv.org/abs/1705.07204

[17] Z. Gong and W. Wang, "Adversarial and Clean Data Are Not Twins," in *Proceedings of the Sixth International Workshop on Exploiting Artificial Intelligence Techniques for Data Management.* New York, NY, USA: ACM, June 2023, pp. 1–5. [Online]. Available: http://arxiv.org/abs/1704.04960https://dl.acm.org/doi/10.1145/3593078.3593935

[18] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial Examples for Malware Detection," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes*

in Bioinformatics), 2017, vol. 10493 LNCS, pp. 62–79. [Online]. Available: http://link.springer.com/10.1007/978-3-319-66399-9_4

[19] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples," *arXiv preprint arXiv:1702.06280*, 2017.

[20] D. Hendrycks and K. Gimpel, "Early Methods for Detecting Adversarial Images," *5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings*, pp. 1–9, August 2016. [Online]. Available: http://arxiv.org/abs/1608.00530

[21] ——, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pp. 1–12, 2019.

[22] H. Hosseini, Y. Chen, S. Kannan, B. Zhang, and R. Poovendran, "Blocking transferability of adversarial examples in black-box learning systems," *arXiv preprint arXiv:1703.04318*, 2017.

[23] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On Detecting Adversarial Perturbations," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pp. 1–12, February 2017. [Online]. Available: http://arxiv.org/abs/1702.04267

[24] D. Miller, Y. Wang, and G. Kesidis, "When not to classify: Anomaly detection of attacks (ADA) on DNN classifiers at test time," pp. 1624–1670, August 2019.

[25] N. Carlini and D. A. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:207599948

[26] M. F. A. Razak, N. B. Anuar, R. Salleh, and A. Firdaus, "The rise of "malware": Bibliometric analysis of malware study," *Journal of Network*

and Computer Applications, vol. 75, pp. 58–76, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804516301904

[27] K. Cabaj and W. Mazurczyk, "Using Software-Defined Networking for Ransomware Mitigation: The Case of CryptoWall," *IEEE Network*, vol. 30, no. 6, pp. 14–20, November 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7764294/

[28] R. Brewer, "Ransomware attacks: detection, prevention and cure," *Network Security*, vol. 2016, no. 9, pp. 5–9, 2016.

[29] P. Zavarsky, D. Lindskog *et al.*, "Experimental analysis of ransomware on windows and android platforms: Evolution and characterization," *Procedia Computer Science*, vol. 94, pp. 465–472, 2016.

[30] Symantec, "Ransomware and businesses 2016," Symantec Corporation, Tech. Rep., 2016, executive Summary. [Online]. Available: https://conferences.law.stanford.edu/cyberday/wp-content/uploads/sites/10/2016/10/5c_ISTR2016_Ransomware_and_Businesses.pdf

[31] K. Gangwar, S. Mohanty, and A. Mohapatra, "Analysis and detection of ransomware through its delivery methods," in *Data Science and Analytics: 4th International Conference on Recent Developments in Science, Engineering and Technology, REDSET 2017, Gurgaon, India, October 13-14, 2017, Revised Selected Papers 4*. Springer, 2018, pp. 353–362.

[32] P. Raunak and P. Krishnan, "Network detection of ransomware delivered by exploit kit," *ARPN Journal of Engineering and Applied Sciences*, vol. 12, no. 12, pp. 3885–3889, 2017.

[33] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X.-y. Zhou, X. Wang *et al.*, "Effective and efficient malware detection at the end host." in *USENIX security symposium*, vol. 4, no. 1, 2009, pp. 351–366.

[34] C. Pascariu and I.-D. Barbu, "Ransomware–an emerging threat," *International Journal of Information Security and Cybercrime*, vol. 4, no. 2, pp. 27–32, 2015.

[35] E. S. Chia, "Singapore's smart nation program — Enablers and challenges," in *2016 11th System of Systems Engineering Conference (SoSE)*. IEEE, June 2016, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/7542892/

[36] R. Rayhana, G. Xiao, and Z. Liu, "Internet of Things Empowered Smart Greenhouse Farming," *IEEE Journal of Radio Frequency Identification*, vol. 4, no. 3, pp. 195–211, September 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9051987/

[37] G. Xu, Y. Peng, W. Che, Y. Lan, W. Zhou, C. Huang, W. Li, W. Zhang, G. Zhang, E. Y. K. Ng, and Y. Cheng, "An IoT-Based Framework of Webvr Visualization for Medical Big Data in Connected Health," *IEEE Access*, vol. 7, pp. 173 866–173 874, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8918431/

[38] I. Bin Aris, R. K. Z. Sahbusdin, and A. F. M. Amin, "Impacts of IoT and big data to automotive industry," in *2015 10th Asian Control Conference (ASCC)*. IEEE, May 2015, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/document/7244878/

[39] N. H. Goddard, *Human Activity Recognition Challenge*, ser. Smart Innovation, Systems and Technologies, M. A. R. Ahad, P. Lago, and S. Inoue, Eds. Singapore: Springer Singapore, 2021, vol. 199. [Online]. Available: http://link.springer.com/10.1007/978-94-015-8935-2_7http://link.springer.com/10.1007/978-981-15-8269-1

[40] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, December 2013. [Online]. Available: http://arxiv.org/abs/1312.6199

[41] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial Machine Learning at Scale," *5th International Conference on Learning Representations, ICLR 2017*

- *Conference Track Proceedings*, pp. 1–17, November 2016. [Online]. Available: http://arxiv.org/abs/1611.01236

[42] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The Limitations of Deep Learning in Adversarial Settings," in *2016 IEEE European Symposium on Security and Privacy (EuroSP)*. IEEE, March 2016, pp. 372–387. [Online]. Available: http://ieeexplore.ieee.org/document/7467366/

[43] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, May 2017, pp. 39–57. [Online]. Available: http://ieeexplore.ieee.org/document/7958570/

[44] C. Xiao, B. Li, J. Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2018-July, pp. 3905–3911, 2018.

[45] X. Han, Y. Hu, L. Foschini, L. Chinitz, L. Jankelson, and R. Ranganath, "Deep learning models for electrocardiograms are susceptible to adversarial attack," *Nature Medicine*, vol. 26, no. 3, pp. 360–363, 2020. [Online]. Available: http://dx.doi.org/10.1038/s41591-020-0791-x

[46] R. K. Sah and H. Ghasemzadeh, "Adar: Adversarial activity recognition in wearables," *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD*, vol. 2019-November, pp. 1–8, 2019.

[47] A. Kurniawan and I. Riadi, "Detection and analysis cerber ransomware using network forensics behavior based." *International Journal Network and Security*, vol. 20, no. 5, pp. 836–843, 2018.

[48] A. Kurniawan, Y. Ohsita, and M. Murata, "Experiments on Adversarial Examples for Deep Learning Model Using Multimodal Sensors," *Sensors*, vol. 22, no. 22, p. 8642, November 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/22/8642

[49] A. Kurniawan, Y. Ohsita, S. Maisuria, and M. Murata, "Detection of sensors used for adversarial examples against machine learning models," *Preprints*, vol. 2023, p. 2023110328, 2023.

[50] A. Kurniawan, Y. Ohsita, and M. Murata, "Toward robust systems against sensor-based adversarial examples based on the criticalities of sensors," in *2024 IEEE 3rd International Conference on AI in Cybersecurity (ICAIC)*. Houston, Texas, USA: IEEE Conference Publications, 2024.

[51] R. Leong, C. Beek, C. Cochin, N. Cowie, and C. Schmugar, "Understanding ransomware and strategies to defeat it," *White Paper (McAfee Labs)*, pp. 1–16, 2016.

[52] McAfee Labs, "Understanding ransomware and strategies to defeat it," Tech. Rep., 2016. [Online]. Available: https://whitepapers.theregister.com/paper/view/4959/understanding-ransomware-and-strategies-to-defeat-it

[53] S. Mansfield-Devine, "Ransomware: taking businesses hostage," *Network Security*, vol. 2016, no. 10, pp. 8–17, 2016.

[54] C. Beek, D. Dinkar, Y. Gund, G. Lancioni, N. Minihane, F. Moreno, E. Peterson, T. Roccia, C. Schmugar, R. Simon *et al.*, "Mcafee labs threats report," *McAfee, Santa Clara, CA, USA, Tech. Rep*, 2017.

[55] A. A. Ahmed and N. A. K. Zaman, "Attack intention recognition: A review." *International Journal Network and Security*, vol. 19, no. 2, pp. 244–250, 2017.

[56] T. Shibahara, T. Yagi, M. Akiyama, D. Chiba, and T. Yada, "Efficient dynamic malware analysis based on network behavior using deep learning," in *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–7.

[57] Sudhakar and S. Kumar, "An emerging threat fileless malware: a survey and research challenges," *Cybersecurity*, vol. 3, no. 1, p. 1, 2020.

[58] M. H. Mate and S. R. Kapse, "Network forensic tool–concept and architecture," in *2015 Fifth International Conference on Communication Systems and Network Technologies*. IEEE, 2015, pp. 711–713.

[59] S. Davidoff and J. Ham, *Network forensics: tracking hackers through cyberspace*. Prentice hall Upper Saddle River, 2012, vol. 2014.

[60] ——, *Network forensics: tracking hackers through cyberspace*. Prentice hall Upper Saddle River, 2012, vol. 2014.

[61] A. Young and M. Yung, "Cryptovirology: Extortion-based security threats and countermeasures," in *Proceedings 1996 IEEE Symposium on Security and Privacy*. IEEE, 1996, pp. 129–140.

[62] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, "Automated dynamic analysis of ransomware: Benefits, limitations and use for detection," *arXiv preprint arXiv:1609.03020*, 2016.

[63] S. S. Ganorkar and K. Kandasamy, "Understanding and defending crypto-ransomware," *ARPN Journal of Engineering and Applied Sciences*, vol. 12, no. 12, pp. 3920–3925, 2017.

[64] A. Provataki and V. Katos, "Differential malware forensics," *Digital Investigation*, vol. 10, no. 4, pp. 311–322, 2013.

[65] J. Li, Q. Li, S. Zhou, Y. Yao, and J. Ou, "A review on signature-based detection for network threats," in *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*. IEEE, 2017, pp. 1117–1121.

[66] P. Szor, "The Art of Computer Virus Research and Defense," 2005. [Online]. Available: https://api.semanticscholar.org/CorpusID:109797805

[67] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," in *The 5th Conference on Information and Knowledge Technology*. IEEE, 2013, pp. 113–120.

[68] S. M. Tabish, M. Z. Shafiq, and M. Farooq, "Malware detection using statistical analysis of byte-level file content," in *Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*, 2009, pp. 23–31.

[69] N. Andronio, S. Zanero, and F. Maggi, "Heldroid: Dissecting and detecting mobile ransomware," in *Research in Attacks, Intrusions, and Defenses: 18th International Symposium, RAID 2015, Kyoto, Japan, November 2-4, 2015. Proceedings 18*. Springer, 2015, pp. 382–404.

[70] J. K. Lee, S. Y. Moon, and J. H. Park, "Cloudrps: a cloud analysis based enhanced ransomware prevention system," *The Journal of Supercomputing*, vol. 73, pp. 3065–3084, 2017.

[71] P. Zavarsky, D. Lindskog *et al.*, "Experimental analysis of ransomware on windows and android platforms: Evolution and characterization," *Procedia Computer Science*, vol. 94, pp. 465–472, 2016.

[72] A. Kharaz, S. Arshad, C. Mulliner, W. Robertson, and E. Kirda, "{UNVEIL}: A {large-scale}, automated approach to detecting ransomware," in *25th USENIX security symposium (USENIX Security 16)*, 2016, pp. 757–772.

[73] N. Zscaler, "White paper: Ransomware is costing companies millions. could it cost you your job," Tech. rep., Zscaler, 110 Rose Orchard Way, San Jose, CA 95134, USA, Tech. Rep., 2016.

[74] A. Kurniawan, I. Riadi, and A. Luthfi, "Forensic analysis and prevent of cross site scripting in single victim attack using open web application security project (OWASP) framework," *Journal of Theoretical and Applied Information Technology*, vol. 95, no. 6, pp. 1363–1371, 2017.

[75] N. Benchikha, M. Krim, K. Zeraoulia, and C. Benzaid, "IWNetFAF: An integrated wireless network forensic analysis framework," in *Proceedings - 2016 Cybersecurity and Cyberforensics Conference, CCC 2016*, 2016.

[76] R. Joshi and E. S. Pilli, *Fundamentals of Network Forensics*. Springer, 2016.

[77] I. Riadi, J. E. Istiyanto, A. Ashari *et al.*, "Log analysis techniques using clustering in network forensics," *arXiv preprint arXiv:1307.0072*, 2013.

[78] M. Baca, J. Cosic, and Z. Cosic, "Forensic analysis of social networks (case study)," in *Proceedings of the ITI 2013 35th International Conference on Information Technology Interfaces*. IEEE, 2013, pp. 219–223.

[79] D. Paul Joseph and J. Norman, "A review and analysis of ransomware using memory forensics and its tools," in *Smart Intelligent Computing and Applications: Proceedings of the Third International Conference on Smart Computing and Informatics, Volume 1*. Springer, 2020, pp. 505–514.

[80] R. Umar, I. Riadi, and R. S. Kusuma, "Analysis of conti ransomware attack on computer network with live forensic method," *IJID (International Journal on Informatics for Development)*, vol. 10, no. 1, pp. 53–61, 2021.

[81] R. S. Kusuma, R. Umar, and I. Riadi, "Network forensics against ryuk ransomware using trigger, acquire, analysis, report, and action (taara) method," *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, 2021.

[82] D. J. Yeong, G. Velasco-hernandez, J. Barry, and J. Walsh, "Sensor and sensor fusion technology in autonomous vehicles: A review," *Sensors*, vol. 21, no. 6, pp. 1–37, 2021.

[83] H. Ichino, K. Kaji, K. Sakurada, K. Hiroi, and N. Kawaguchi, "HASC-PAC2016: large scale human pedestrian activity corpus and its baseline recognition," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and*

Ubiquitous Computing: Adjunct. New York, NY, USA: ACM, September 2016, pp. 705–714. [Online]. Available: https://dl.acm.org/doi/10.1145/2968219.2968277

[84] I. Debache, L. Jeantet, D. Chevallier, A. Bergouignan, and C. Sueur, "A lean and performant hierarchical model for human activity recognition using body-mounted sensors," *Sensors (Switzerland)*, vol. 20, no. 11, 2020.

[85] C. Benegui and R. T. Ionescu, "Adversarial attacks on deep learning systems for user identification based on motion sensors," in *Neural Information Processing: 27th International Conference, ICONIP 2020, Bangkok, Thailand, November 18–22, 2020, Proceedings, Part V 27*. Springer, 2020, pp. 752–761.

[86] R. Kumar Sah and H. Ghasemzadeh, "Adversarial Transferability in Wearable Sensor Systems," *arXiv*, vol. 1, no. 1, pp. 1–23, 2020.

[87] S. Jandial, P. Mangla, S. Varshney, and V. Balasubramanian, "AdvGAN++: Harnessing latent layers for adversary generation," *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, pp. 2045–2048, 2019.

[88] A. Liu, X. Liu, J. Fan, Y. Ma, A. Zhang, H. Xie, and D. Tao, "Perceptual-Sensitive GAN for Generating Adversarial Patches," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 1028–1035, July 2019. [Online]. Available: https://aaai.org/ojs/index.php/AAAI/article/view/3893

[89] Y. Kim, H. Kang, N. Suryanto, H. T. Larasati, A. Mukaroh, and H. Kim, "Extended spatially localized perturbation gan (Eslp-gan) for robust adversarial camouflage patches†," *Sensors*, vol. 21, no. 16, pp. 1–18, 2021.

[90] W. Hackett, S. Trawicki, Z. Yu, N. Suri, and P. Garraghan, "Pinch: An adversarial extraction attack framework for deep learning models," *arXiv preprint arXiv:2209.06300*, 2022.

[91] O. Banos, R. Garcia, J. A. Holgado-Terriza, M. Damas, H. Pomares, I. Rojas, A. Saez, and C. Villalonga, "mHealthDroid: A Novel Framework for Agile

Development of Mobile Health Applications," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8868, no. January, pp. 91–98. [Online]. Available: http://link.springer.com/10.1007/978-3-319-13105-4_14

[92] R. Mutegeki and D. S. Han, "A CNN-LSTM Approach to Human Activity Recognition," in *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. IEEE, February 2020, pp. 362–366. [Online]. Available: https://ieeexplore.ieee.org/document/9065078/

[93] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[94] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," *34th International Conference on Machine Learning, ICML 2017*, vol. 7, pp. 5109–5118, 2017.

[95] Y. Senzaki, S. Ohata, and K. Matsuura, "Simple black-box adversarial examples generation with very few queries," *IEICE Transactions on Information and Systems*, vol. E103D, no. 2, pp. 212–221, 2020.

[96] P. Karle, F. Fent, S. Huch, F. Sauerbeck, and M. Lienkamp, "Multi-Modal Sensor Fusion and Object Tracking for Autonomous Racing," *IEEE Transactions on Intelligent Vehicles*, pp. 1–13, 2023.

[97] X. Zhang, P. Zheng, T. Peng, D. Li, X. Zhang, and R. Tang, "Privacy-preserving activity recognition using multimodal sensors in smart office," *Future Generation Computer Systems*, vol. 148, pp. 27–38, may 2023.

[98] H. Zhou, Y. Zhao, Y. Liu, S. Lu, X. An, and Q. Liu, "Multi-Sensor Data Fusion and CNN-LSTM Model for Human Activity Recognition System," *Sensors*, vol. 23, no. 10, p. 4750, May 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/10/4750

[99] L. Yuan, J. Andrews, H. Mu, A. Vakil, R. Ewing, E. Blasch, and J. Li, "Interpretable passive multi-modal sensor fusion for human identification and activity recognition," *Sensors*, vol. 22, no. 15, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/15/5787

[100] F. Tramèr and D. Boneh, "Adversarial training and robustness for multiple perturbations," in *Advances in Neural Information Processing Systems*, vol. 32, no. NeurIPS, 2019, pp. 1–11.

[101] D. Kang, Y. Sun, T. Brown, D. Hendrycks, and J. Steinhardt, "Transfer of adversarial robustness between perturbation types," *arXiv preprint arXiv:1905.01034*, 2019.

[102] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfittin," *Journal of Machine Learning Research 15*, vol. 15, pp. 1929–1958, January 2014. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/037026939390272J