INVITED PAPER

Edge Computing Security: State of the Art and Challenges

This article reviews the most influential and basic attacks as well as the corresponding defense mechanisms that can be practically applied in edge computing systems.

By Yinhao Xiao, Yizhen Jia, Chunchi Liu, Xiuzhen Cheng, *Fellow IEEE*, Jiguo Yu, *Senior Member IEEE*, and Weifeng Lv

ABSTRACT | The rapid developments of the Internet of Things (IoT) and smart mobile devices in recent years have been dramatically incentivizing the advancement of edge computing. On the one hand, edge computing has provided a great assistance for lightweight devices to accomplish complicated tasks in an efficient way; on the other hand, its hasty development leads to the neglection of security threats to a large extent in edge computing platforms and their enabled applications. In this paper, we provide a comprehensive survey on the most influential and basic attacks as well as the corresponding defense mechanisms that have edge computing specific characteristics and can be practically applied to real-world edge computing systems. More specifically, we focus on the following four types of attacks that account for 82% of the edge computing attacks recently reported by Statista: distributed denial of service attacks, side-channel attacks, malware injection attacks, and authentication and authorization attacks. We also analyze the root causes of these attacks, present the status quo and grand challenges in edge computing security, and propose future research directions.

KEYWORDS | Data security; edge computing; Internet of Things; network security.

Manuscript received February 13, 2019; revised April 18, 2019; accepted May 13, 2019. Date of publication June 19, 2019; date of current version August 5, 2019. This work was supported in part by the National Natural Science Foundation of China under Grants U1811463, 61832012, 61771289, 61672321, and 61373027; and in part by the U.S. National Science Foundation under Grants CNS-1704397 and IIS-1741279. *(Corresponding author: Jiguo Yu.)*

Y. Xiao is with the School of Computer Science and Technology, Shandong University, Qingdao 266237, China, with the Department of Computer Science, The George Washington University, Washington, DC 20052 USA, and also with the School of Information Science, Guangdong University of Finance and Economics, Guangzhou 510320, China (e-mail: xyh3984@gwu.edu).

Y. Jia, **C. Liu**, and **X. Cheng** are with the School of Computer Science and Technology, Shandong University, Qingdao 266237, China, and also with the Department of Computer Science, The George Washington University, Washington, DC 20052 USA (e-mail: chen2015@gwu.edu; liuchunchi@gwu.edu; cheng@gwu.edu).

J. Yu is with the School of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250353, China, with the Shandong Computer Science Center (National Supercomputer Center in Jinan), Jinan 250014, China, and also with the School of Information Science and Engineering, Qufu Normal University, Rizhao 276826, China (e-mail: jiguoyu@sina.com).

W. Lv is with the State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China, and also with the Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing 100191, China (e-mail: lwf@nlsde.buaa.edu.cn).

Digital Object Identifier 10.1109/JPROC.2019.2918437

I. INTRODUCTION

Because of the smart city boom in recent years, the number of deployed mobile and Internet of Things (IoT) devices has been growing drastically. These devices are adopted as the basic components in smart city infrastructures to undertake the most fundamental but indispensable tasks such as sensing, actuating, and controlling. However, relying only on the devices alone is insufficient to fully accomplish sophisticated tasks such as smart transportation arrangements and smart medical treatments. In this case, a high-performance computing platform is needed for these IoT/mobile devices to offload their computation tasks and assist them in making decisions. The most well-known of such technology is cloud computing. Yet, traditional cloud computing, which is used to support general computing systems, can hardly satisfy the needs of IoT and mobile services due to reasons such as location unawareness, bandwidth shortage, operation cost imposition, lack of real-time services, and lack of data privacy guarantee.

These limitations of cloud computing pave the way for the advent of edge computing, a technology that is believed to be able to cope with the demands of the ever-growing IoT and mobile devices. The basic idea of edge computing is to employ a hierarchy of edge servers with increasing computation capabilities to handle mobile and heterogeneous computation tasks offloaded by the low-end IoT and mobile devices, namely, edge devices. Edge computing has the potential to provide location-aware, bandwidth-sufficient, real-time, privacy-savvy, and low-cost services to support emerging smart city applications. Such advantages over cloud computing have caused edge computing to grow rapidly in recent years. According to the latest report by Statista, the market size of edge computing in the United States was projected to reach \$1031 million by 2025 from the current \$84.3 million in 2018 [1]. According to another recent report, the total number of IoT devices that were in use worldwide reached 11.2 billion in 2018 and was projected to reach 20 billion by 2020 [2].

On the one hand, edge computing provides a more feasible computing technology for smart city applications and beyond; on the other hand, its emergence introduces more security threats since it increases the real-world attack surface from the following four angles.

1) *Weak Computation Power:* Compared to a cloud server, the computation power of an edge server is relatively weaker. Therefore, an edge server is more vulnerable to existing attacks that may no longer be effective against a cloud server. Similarly, compared to general-purpose computers, edge devices have more fragile defense systems; as a consequence, many attacks that may be ineffective against desktop computers can pose serious threats to edge devices.
2) *Attack Unawareness:* Unlike general-purpose computers, majority of IoT devices do not have user interfaces (UIs), regardless of the fact that some may have crude light-emitting diode (LED) screens. Therefore, a user may have limited knowledge about the running status of a device, e.g., whether it has been shut down or compromised. Hence, even if an attack is taking place in an edge device, most users may not be able to discern it.
3) *OS and Protocol Heterogeneities:* Unlike general-purpose computers that tend to use standard OSes and communication protocols such as POSIX [3], most edge devices have different OSes and protocols without a standardized regulation. This problem directly leads to the difficulties of designing a unified protective mechanism for edge computing.
4) *Coarse-Grained Access Control:* The access control models designed for the general-purpose computers and cloud computing mainly consist of four types of permissions: No Read & Write, Read Only, Write Only, and Read & Write [4]. Such a model would never be satisfiable in edge computing due to the more complicated systems and their enabled applications, which call for fine-grained access control that should handle questions such as "who can access which sensors by doing what at when and how." Unfortunately, current access control models are mostly coarse-grained [5].

Accordingly, the attacks targeting edge computing infrastructures have a drastic rise in recent years. One of the most notable attacks that happened in the real world is the Mirai virus, which managed to compromise more than 65 000 IoT devices within the first 20 h after its release in August 2016 by exploiting the devices' weak authentication vulnerabilities [6]. A few days later, these compromised devices were turned into botnets to launch distributed denial of service (DDoS) attacks against edge servers, shutting down over 178 000 domains [7]. Shortly after that, variations of Mirai, such as IoTReaper and Hajime, were captured, and they were believed to infect more than 378 million IoT devices in 2017 [6]. Since the discovery of the first Mirai botnet in 2016, IoT botnet attacks were reported to cause damages worthy of over \$100 million by September 2018 [8]. Note that these numbers only indicate the attacks and asset losses that were officially identified and reported, and the total amounts of undetected attacks/losses could be much higher.

The Mirai example described earlier demonstrates the dire straits in edge computing security. In this paper, we provide a comprehensive survey on practical state-of-the-art attacks and defense solutions proposed for edge computing systems. Major attacks that can be directly applied to edge computing applications are classified into six categories, namely, DDoS attacks, side-channel attacks, malware injection attacks, authentication and authorization attacks, man-in-the-middle attacks, and bad-data injection attacks. The percentage of each class of attacks happened in 2017 targeting real-world edge computing infrastructures is shown in Fig. 1, according to the most recent report posted by Statista [9]. Note that the total number of IoT attacks discovered in 2017 is 159 700 [10], with almost all falling into these six categories. Nevertheless, this paper covers in depth only the first four for the following two reasons: 1) it is impossible to exhaustively survey all possible attacks, and this paper focuses only on the most influential and basic ones that can be practically launched in real-world edge computing systems; and 2) the man-in-the-middle attacks reported in the context of edge computing do not carry sufficient uniqueness compared with their counterparts in cloud computing and Internet computing, while bad-data injection attacks mainly happened in smart grids, whose generalization to general edge computing systems is quite limited. We also summarize the root causes of the attacks, outline the status quo and grand challenges of edge computing security, and propose future research directions.

The remaining of this paper is organized as follows. In Section II, we present the basic architecture of edge computing. In Section III, we demonstrate the state-of-the-art security attacks and defense mechanisms that can be practically implemented in edge computing systems. Section IV outlines the root causes of edge computing

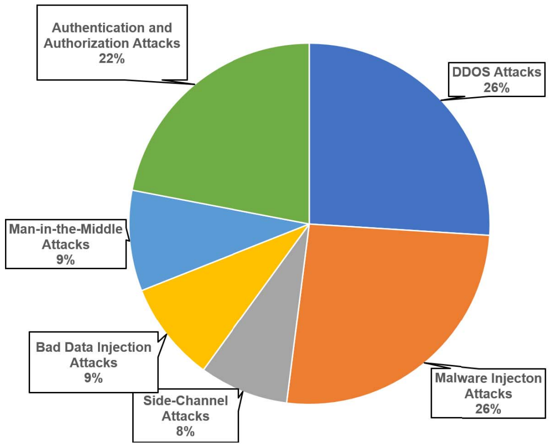


Fig. 1. ***Percentages according to the types of attacks targeting edge computing infrastructures happened in real world.***

security threats, presents the status quo and grand challenges in securing edge computing systems, and proposes the future research directions. Finally, we conclude this paper in Section V.

II. ARCHITECTURE OF EDGE COMPUTING

In this section, we present a general architecture of edge computing shown in Fig. 2, which mainly consists of three layers: an edge device layer (EDL), an edge server layer (ESL), and a cloud server layer (CSL). From the perspective of computational power, the systems built on the CSL have the most powerful computation capability, followed by the ones on the ESL. The devices at the EDL usually have the lowest computational power.

A. Edge Device Layer

Edge devices are those low-level electronic devices deployed at EDL which operate in the physical world to complete tasks such as sensing, actuating, and controlling. Each edge device is logically controlled by one or more microcontrollers (MCUs), with each being a small computer running on a single integrated circuit [11]. The low-level software interface programmed in the MCUs that provide controls to the device's hardware is known as firmware. All the functions, including sensing, controlling, and computing, are coded in the firmware and, therefore, handled by the MCUs. Edge devices can be further categorized as IoT devices and mobile devices. IoT devices are lightweight electronic devices that are interconnected or connected to the edge servers in ESL through wireless protocols such as 4G/5G, Wi-Fi, and Bluetooth. They usually run on lightweight preemptive/cooperative real-time operating systems (RTOSs), e.g., FreeRTOS and RT Thread [12] [13]. Once an RTOS is burned into the chip of the IoT devices, it usually does not provide further programming interfaces. Some examples of IoT devices include smart home devices, health monitoring devices, and smart warehouse carts in industrialized IoT (IIoT). Most of the manufacturers of IoT devices adopt Cortex-M series MCUs produced by STMicroelectronics [14]. Different from IoT devices, mobile devices usually have more advanced and costly preemptive operating systems, e.g., Android and iOS, providing programmable interfaces for developers to code their own applications at the top of the OSes. Some examples of mobile devices include smartphones, tablets, and central controllers of smart vehicles. Most of the manufacturers of mobile devices adopt Cortex-A series MCUs produced by high-performance chip manufacturers such as Qualcomm [15].

B. Edge Server Layer

ESL has a hierarchical structure with multiple sublayers consisting of various edge servers with increasing computational power from bottom to up, as shown in Fig. 2. The edge servers located at the lowest sublayer include wireless base stations and access points (APs), which are mainly deployed for communication purpose to receive data from the edge devices and send control flows back to them through different wireless interfaces. Upon receiving data from edge devices, base stations/APs forward the data to the edge servers located at the upper sublayer, which are mainly in charge of handling computation tasks. Upon receiving data passed from base stations/APs or edge servers at the lower sublayers, the edge servers conduct relevant computation and analysis tasks on their own. If the complexity of a task exceeds the computation limits of the current edge server, it would offload the task to the servers located at the higher sublayers, which possess more powerful computation capabilities. These servers then conclude with a sequence of control flows and pass them back to the base stations/APs, which forward them to the edge

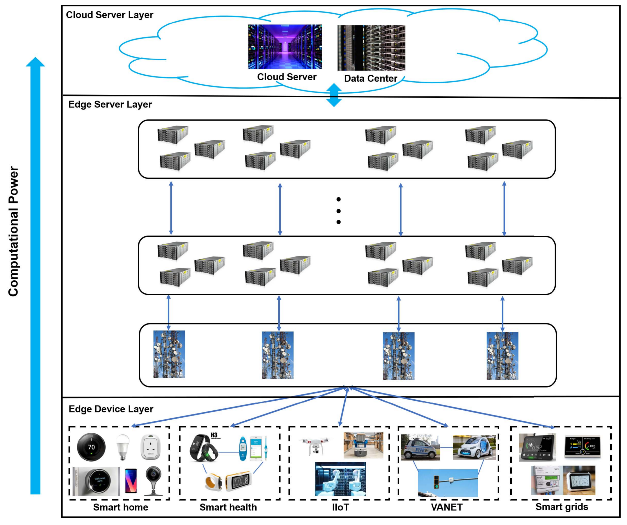


Fig. 2. ***General architecture of edge computing.***

devices in the end. Edge servers handle most of the core computing functions such as authentication, authorization, computation, data analytics, task offloading, and data storage for edge computing.

C. Cloud Server Layer

CSL hosts center cloud servers and data centers, with the cloud servers responsible for the highest level authentication and authorization, computation, and integration of different tasks offloaded from edge servers, and the data centers in charge of storing a vast amount of data generated by the edge devices and edge servers. The state-of-the-art cloud servers and cloud data centers consist of clusters of powerful machines. Because the security of CSL has been extensively studied [16], [17], in this paper, we mainly explore the security issues of EDL and ESL in edge computing.

III. EDGE COMPUTING SECURITY AND PRIVACY

In this section, we first summarize the major state-of-the-art security threats and attacks faced by edge computing. These threats have mainly resulted from the design flaws, misconfigurations, and implementation bugs. We then explore the corresponding defense mechanisms that are either detection based for the purpose of recognizing attacks from normal activities or prevention based for the objective of obstructing the attacks from happening. In many cases, detection-based solutions may be the only choice, especially for defending the attacks that exploit the design flaws within a system since such flaws can only be fixed based on a detection-patch logic. Finally, we outline the root causes of the attacks and discuss the practicality of launching/defending them.

A. DDoS Attacks and Defense Mechanisms

DDoS refers to a type of cyberattack in which attackers aim to disrupt normal services provided by one or more servers based on distributed resources such as a cluster of compromised edge devices (also known as botnet) [18]. It is a powerful attack that aims to prevent the legitimate use of a service. A traditional DDoS attack occurs when an attacker persistently sends streams of packets to a victim from the compromised distributed electronic devices; thus, the hardware resources of the victim are quickly exhausted for handling these malicious packets and can no longer process any legitimate request on time. In some other DDoS scenarios, an attacker persistently sends malformed packets that confuse an application or a protocol of the victim to falsely conclude that all channels and resources are occupied. Compared with cloud servers, edge servers are more susceptible to DDoS attacks since they are relatively computationally less powerful to maintain strong defense systems as cloud servers do. In addition, edge servers mainly provide services to edge devices that are well-known to be error-prone in regard to security

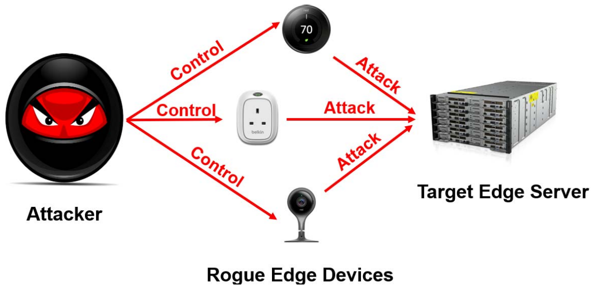


Fig. 3. *Typical architecture of the DDoS attack.*

settings due to their computation-limited hardware and heterogeneous firmware. Having noticed this, attackers favor first compromising a number of edge devices and turning them into weapons against edge servers. The Mirai botnet is an infamous example where the attacker took control of over 65 000 IoT devices within the first 20 h after its release. These rogue IoT devices were then exploited to launch a DDoS attack targeting high-profile edge service providers such as Krebs, OVH, and Dyn [6]. Shortly after the outbreak of Mirai, several variation botnets, e.g., Hajime and BrickerBot, were unearthed [18]. DDoS is the most commonly adopted and easiest-to-exploit attack in the practical world, as shown in Fig. 1. Hence, it poses a significant threat to the real-world edge computing services.

1) Attack Specifications: DDoS attacks may happen when malicious edge devices communicate with the edge servers. In a DDOS attack, an attacker first compromises a cluster of edge devices and takes full control of them; then, it commands each device to launch a denial-of-service attack targeting the edge server, causing the shutdown of its services. A typical architecture of a DDoS attack in edge computing is shown in Fig. 3. DDoS attacks targeting edge computing can be taxonomized as flooding-based attacks and zero-day attacks.

a) Flooding-based attacks: Flooding-based attacks are a type of DDoS attacks aiming to shut down the normal service of a server based on a large amount of flooded malformed/malicious network packets and are mainly classified as UDP flooding, ICMP flooding, SYN flooding, ping of death (PoD), HTTP flooding, and Slowloris, according to the attack techniques. In a UDP flooding attack, an attacker continuously sends a large amount of noisy UDP packets to a target edge server, causing the server incapable of handling the benign UDP packets in time and thus interrupting the normal UDP services provided by the edge server [19]. In an ICMP flooding attack, an attacker exploits the ICMP protocol to craft an attack by sending a large number of ICMP Echo Request packets to a target edge server as fast as possible without waiting for the replies. This type of attack consumes both outgoing and incoming throughputs of the victim server since the server returns an ICMP Echo Reply packet upon each receipt of a ping request, resulting in a significant system-wide slowdown [20]. In a SYN flooding attack, an attacker exploits the three-way handshake of the transmission control protocol (TCP) by initiating a huge amount of SYN requests with a spoofed IP address to a target edge server, while the server responds with a SYN-ACK packet to the spoofed IP for each SYN request and waits for the confirmation ACK that never comes [21]. In a PoD attack, an attacker creates an IP packet with malformed or malicious content that has a length greatly larger than the maximum frame size for a standard IP packet (65 535 bytes) and splits the long IP packet into multiple fragments and sends them to a target server. Upon receipts of the fragments, the server must reassemble all the fragments that end up with an IP packet whose size is over 65 535 bytes. If the attacker continuously sends a large number of such packets to the target, the computation resources of the target server may be completely occupied to reassemble all the fragments [22]. In an HTTP flooding attack, an attacker simply sends a tremendous amount of HTTP `GET`, `POST`, `PUT`, or other legitimate requests to an edge server. Since the edge server is less likely able to handle a huge number of legitimate requests in a timely manner due to the restricted computation power, its normal services can be easily throttled if a large amount of HTTP traffics are received in a short period of time [23]. In a Slowloris attack, an attacker creates numerous partial HTTP connections, which, for example, can be realized by only sending HTTP headers to a target server but never completing one. In this case, the server keeps all the open connections falsely in different concurrent threads/processes until the total number hits the maximum concurrent connection pool size, causing the shutdown of the server [24].

b) Zero-day DDoS attacks: A zero-day DDOS attack is more advanced than flooding-based DDoS mentioned earlier, but it is more difficult to implement. In such an attack, an attacker must find an unknown vulnerability (i.e., zero-day vulnerability) in a piece of code running on the target edge server/device, which can cause memory corruption and finally result in a service shutdown. For instance, the common vulnerabilities and exposures (CVE)-2010-3972 is a heap-based overflow that can cause a DoS on Internet Information Services (IIS) 7.0 and IIS 7.5 [25]. This kind of attack is also the most difficult one to defend against since it exploits a zero-day vulnerability that has not been known to the public.

2) Current Defense Solutions: The root cause of the flooding-based attacks is the protocol-level design flaws/vulnerabilities within the network communication protocols, while the root cause of the zero-day attacks lies in the code-level vulnerabilities that can trigger memory failures/corruptions. Accordingly, current defense solutions against flooding-based attacks mainly adopt a detect-filter philosophy, while those against the zero-day attacks mainly focus on code-level vulnerability identification.

a) Defense solutions against flooding-based attacks: Detection of the flooding-based DDoS attacks can be mainly classified into two categories: per-packet-based detection and statistics-based detection.

Per-packet-based detections aim to detect flooding-based attacks at the packet level. Intuitively, since a flooding-based DDoS attack is launched mainly by sending an enormous amount of malicious or malformed network packets, detecting and filtering those packets can have an effective defense. This observation was exploited in [26], which proposed integrating packet filtering mechanisms into congestion control frameworks to mitigate the attacks. When a suspicious packet is identified, the network can simply drop the packet before it arrives at the destined edge server. Although this may sound trivial, detecting whether a packet is a DoS-oriented malicious one is never easy. Attackers can leverage advanced techniques such as spoofing the packets using counterfeit IP/MAC addresses and choosing carefully crafted HTTP headers and agents [27] to make the DDoS attacks more stealthy. Therefore, researchers turned to develop more effective detection schemes. Yaar *et al.* [28] proposed a mechanism to spot DDoS on a per-packet basis. Their method mainly exploits the fact that packets coming from the same path have the same identifier. Hence, if a packet has the same identifier as a DDoS packet previously spotted, it is highly likely that this packet is also DDoS-oriented. Similarly, Luo *et al.* [29] investigated techniques by detecting possible DDoS packets based on packet identifiers. However, a more sophisticated attacker can easily circumvent such detection mechanisms by changing the identifiers of the packets using tools such as `hping3` [30]. Xu *et al.* [31] then proposed a negative selection algorithm to quickly figure out whether the IP address of a packet is legitimated based on the eigenvalue sets to resist this type of DDoS. Nevertheless, this solution approach requires the server to maintain a list of legitimate IP addresses, which implies that if a client changes its IP address, it has to report the change to the server, making the whole process less efficient.

Statistics-based approaches mainly detect DDoS attacks based on the advent of clusters of DDoS traffics. The advantage of such methods lies in that they do not require the per-packet information such as packet identifier and IP/MAC addresses for attack detection. The existing statistics-based detection solutions employ either packet entropy or machine learning tools. Researchers developed various entropy-based mechanisms to detect possible DDoS traffics [32]–[34]. These methods not only more or less require manual efforts, which may face great challenges if the DDoS traffics are encrypted, but also need the distribution of a large amount of traffics to achieve an accurate detection. To automate the detection, researchers switched to seek possible solutions from machine learning and deep learning techniques. Livadas *et al.* [35] leveraged basic machine learning methods such as J48, naive Bayes, and Bayesian network classifiers to detect botnet DDoS. Zolotukhin *et al.* [36] proposed a deep learning model using an autoencoder to detect encrypted DDoS traffics. Niyaz *et al.* [37] used neural networks to identify DDoS attacks in software-defined networks. Although learning-based mechanisms require little human effort, they are susceptible to overfitting, meaning that they may perform differently on different types of DDoS attacks. To sum up, one can see that statistics-based detection needs a large amount of DDoS traffics for entropy computation or training purpose, which implies that such detection mechanisms can only begin to function after a fair amount of DDoS traffics already damage the edge servers.

b) Defense solutions against zero-day attacks: To defend against this type of attacks, researchers developed mechanisms such as pointer taintedness detection [38] and ECC-memory [39] to spot possible memory leaks in a program. Yet, such methods require the presence of the original source codes, which are usually unavailable for edge devices. Later, Shoshitaishvili *et al.* [40], Gupta and Shenoy [41], and Muench *et al.* [42] proposed the approaches to perform memory analysis based solely on firmware. Chua *et al.* [43], Song *et al.* [44], and Zuo *et al.* [45] showed that with the help of deep learning models, e.g., recurrent neural networks (RNNs), graph neural networks (GNNs), and deep natural language processing (NLP), one can identify vulnerabilities in firmware with higher accuracy rates. Nevertheless, the firmware is usually not available nowadays due to encryption and antidebug fuses [46]. On the other hand, discovering and fixing memory corruptions when the firmware is unavailable is nontrivial. The only work under our attention is the one by Chen *et al.* [47] who demonstrated fuzzing mechanisms from the IoT-app side to identify possible memory corruptions without the need of firmware. However, this solution requires constant human interactions, making it unscalable and infeasible in some scenarios. Besides mechanisms associated with vulnerability discoveries, researchers also sought ways to actively protect edge devices from zero-day attacks. Frassetto *et al.* [48] proposed an in-process memory isolation extension module to the binary to defend against possible memory corruption attacks. Nevertheless, their method was designed specifically for x86 platforms and might consume extra computing resources, making it less feasible to be adopted in most resource-constrained IoT devices. Shirali-Shahreza and Ganjali [49] proposed an IoT firewall using software-defined networking (SDN) to reduce the attack surface of an exposed IoT device. Dietz *et al.* [50] deployed lightweight isolation mechanisms on access routers that serve as guards before an IoT botnet virus can access real edge devices. Note that even though these mechanisms can provide active protections, they cannot fix the zero-day vulnerabilities and hence may be ineffective if a zero-day vulnerability is too complex to trigger or has not been discovered before.

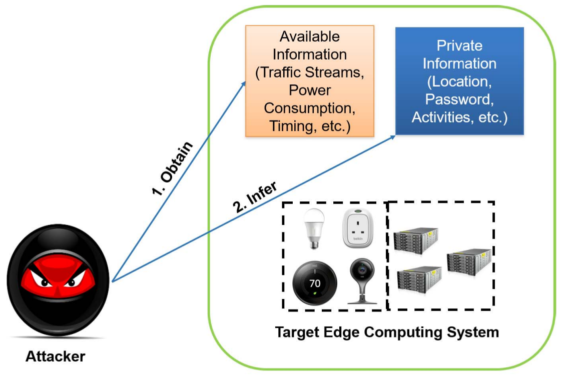
3) Discussion: Based on the overviews on DDoS attacks and defenses presented in Section III-A, one can see that the root cause of flooding-based DDoS attacks is the protocol-level flaws caused by the neglect of security in the initial design, while the root cause of zero-day attacks is the code-level vulnerabilities that can trigger memory failures/corruptions. Flooding-based attacks are easy to launch in the real world as attackers can simply create a large number of malicious packets from the compromised distributed devices, while zero-day attacks may not be as common since discovering zero-day vulnerabilities in a system requires extremely sophisticated analysis. Nevertheless, it is still quite practical to launch a zero-day attack in practice since code-level vulnerabilities are difficult to avoid when developers program for a large system with millions of lines of codes. Such vulnerabilities are even harder to avoid in edge computing as edge devices usually adopt partially fledged system software to trade off stronger security with lower cost and better user experience.

Current defenses against DDoS attacks are still very limited. Per-packet-based detection of flooding-based attacks either can be bypassed by more sophisticated DDoS attacks that exploit address/identifier spoofing or need to maintain a large list of legitimate IP addresses that may be subjected to frequent changes; statistics-based detection, on the other hand, identifies DDoS attacks only after groups of DDoS packets have already been sent to the target edge servers, causing irreparable damages. Zero-day attacks are even harder to defend as most offline detection mechanisms cannot pinpoint the exact type or location of the vulnerabilities, and most real-time online defense systems can hardly figure out the attacks if the DDoS attack shellcodes are encrypted or deliberately modified.

B. Side-Channel Attacks and Defenses

Side-channel attacks refer to those that compromise a user's security and privacy using any publicly accessible information that is not privacy-sensitive in nature, namely, side-channel information. Such public information is typically correlated "secretly" with certain privacy-sensitive data that should be protected. Attackers then explore the hidden correlations to finally infer the protected data from the side channels. Since any public information can have the potential to link to some sensitive data, side-channel attacks can happen anywhere in the edge computing architecture.

1) Attack Specifications: A typical architecture of side-channel attacks is shown in Fig. 4. An attacker constantly obtains certain side-channel information from the target edge computing infrastructure and then feeds it into specific algorithms or machine learning models that output the desired sensitive information. The most popular side channels in edge computing include communication signals, electric power consumption, and smartphone `/proc` filesystem or embedded sensors. The attacks exploiting communication channels happen when the attacker continuously monitors the transmissions between two edge



Available Information (Traffic Streams, Power Consumption, Timing, etc.)
Private Information (Location, Password, Activities, etc.)
1. Obtain
2. Infer
70
Attacker
Target Edge Computing System

Fig. 4. *Typical architecture of the side-channel attack.*

nodes, those exploiting power consumption occur after the attacker steals power consumption data of the edge devices, and those exploiting smartphone-based channels happen when the attacker secretly accesses the smartphone and steals the information stored in the publicly available `/proc` file or generated by the embedded sensors.

a) Attacks exploiting communication channels: In edge computing, exploiting communication signals has a high potential to reveal sensitive information of a victim due to the rich channel information. In this case, an attacker can be any curious malicious node, which does not have to be an edge device or an edge server, who continuously sniffs the network traces and wishes to extract sensitive information out of them. We can further taxonomize this type of attacks into two subclasses: those exploiting packet streams and those exploiting wave signals.

A packet is the atomic unit in most communication channels. A sequence of packets contain rich information and, hence, are widely exploited by attackers to infer sensitive data. Li *et al.* [51] showed that the different coding scheme employed by H.264 and MPEG-4 to reduce temporal redundancy in adjacent video frames can cause severe privacy leak in home surveillance even if the video stream is encrypted. They found that leveraging simple machine learning algorithms such as k-nearest neighbors (k-NN) and density-based spatial clustering of applications with noise (DBSCAN) can achieve an accuracy as high as 95.8% for inferring the four standard human daily activities (dressing, styling hair, moving, and eating) defined by the Health Insurance Portability and Accountability Act (HIPAA). Ji *et al.* [52] adopted an inference model that takes the long short-term memory (LSTM) deep learning network as an attack vector to detect the existence of wireless cameras and to infer the user presence around a target house. They successfully achieved a prediction accuracy of 97.2% as a result. Researchers also demonstrated that inference attacks can be launched by exploiting the IoT traffic streams. Apthorpe *et al.* [53] monitored the encrypted IoT traffics and developed a three-step attack by first separating the traffic into individual device flows using the IP addresses of the edge servers, then correlating each flow with its responsible IoT device according to unique identifiers, and finally inferring user activities from the traffic rate changes. Chen and Qian [54] demonstrated that due to a subtle timing channel vulnerability introduced by most wireless routers that respond to different TCP packets with different timing gaps, an attacker can easily infer the correct TCP packet number and conduct off-path TCP packet injection attacks.

Wave signals are another type of side channels existing in a communication process and may have the potential to reveal a victim's sensitive information. One of the notable examples is the electromagnetic interference (EMI). Enev *et al.* [55] carried out an attack to infer the video content playing in modern TVs through the discernible EMI signatures. Selvaraj *et al.* [56] showed that with the help of intentional EMI (IEMI), an attacker can manipulate the input and output signals of an IoT sensory device from the physical layer and bypass the traditional integrity checking mechanisms. Besides EMI, researchers also found that Wi-Fi waves can be used as side channels to conduct inference attacks. Li *et al.* [57] proposed a malware-less side-channel attack by exploiting channel state information (CSI) to infer a victim's sensitive password input such as Alipay code based on the finger movements. Existing studies also showed that human brain wave data contain rich information that can be used to conduct inference attacks. A group of researchers from the University of Oxford studied the privacy threat from brain–computer interfaces (BCIs) [58] and showed that if an attacker can successfully capture the raw electroencephalography (EEG) data (i.e., human brain wave data), combined with machine learning algorithms such as boosted logistic regression, stepwise linear discriminant analysis (SWLDA), and Fisher's linear discriminant analysis (FLDA), the attacker can infer victim's banking information, month of birth, face, and geographic location with the accuracies of 15%–40% better than the random guessing attack. Xiao *et al.* [59] demonstrated that an attacker can still infer a user's fine-grained activities even with reduced-featured EEG data.

b) Attacks exploiting power consumption: Power consumption is an indicator of the electric usage of a system. It carries information related to either the device that consumes the energy as different devices have different power consumption profiles when operating or the intensity of computations in a computing task. Hence, it attracts attention from a number of researchers to investigate its link to sensitive data. We further categorize this type of attacks into two subclasses: attacks exploiting power consumption collected by meters and those exploiting power consumption collected by oscilloscopes.

Smart meters can accurately measure the electric power consumption of a household. Therefore, such data can be exploited to infer sensitive household activities. In as early as 1992, Hart [60] proposed a side-channel inference method named nonintrusive appliance load monitoring (NILM) to monitor simple device states, e.g., ON or OFF, based on the energy consumption of individual appliances.

This inference was benign as it was not employed for malicious attacks. Later, Stankovic *et al.* [61] revised the original NILM to conduct an inference attack and showed that most household activities, such as cooking, washing, laundering, watching TV, gaming, and so on, can be inferred from the energy data available in a smart meter infrastructure. Clark *et al.* [62] carried out a side-channel attack leveraging the power outlet of an edge device and managed to infer the webpage that the device was visiting with approximately 99% of accuracy. Later, they extended the work to show that using the power energy as an inference feature can even detect malicious malware in an edge device with an accuracy of approximately 94% [63]. Even though this paper is not attack-oriented, it indirectly shows that the sensitive behaviors of benign software can also possibly be identified based solely on power consumption. Recently, researchers found a new physical channel, a thermal side-channel caused by power consumption, and exploited it to more effectively conduct time power attacks and hence compromising the edge data center availability [64].

An oscilloscope is an instrument measuring the electronic information (e.g., voltage and current) of a hardware device. In modern embedded devices, some chip can perform complicated cryptographic algorithms such as AES-CCM, with a hardcoded secret key in the chip. Such a secret key cannot be directly cracked if no software-level vulnerabilities are present. However, researchers found that the power consumption of the hardware may be susceptible to leaking the key. Örs *et al.* [65] demonstrated that adopting simple power analysis and differential power analysis can reveal a significant amount of information carried by the elliptic curve cryptosystem on a field-programmable gate array (FPGA) chip. Ronen *et al.* [46] showed that adopting correlation power analysis can completely reverse the AES-CCM master key used to encrypt/decrypt the firmware installed in the Philips hue smart lights such that they can deliberately create any malicious firmware and install on any Philips hue smart light over-the-air. What is even worse is that as reported in [66], nearly all cryptographic approaches and their corresponding hardware are vulnerable to power analysis attacks. Yet, launching power analysis attacks requires an attacker to be able to physically access the target device or through malicious apps, making this class of attacks difficult to implement in practice.

c) Attacks exploiting smartphone-based channels: Smartphones are key edge devices in many applications. Different from IoT devices, smartphones have more advanced OSes and possess richer system information. Therefore, compared with the less advanced IoT devices, smartphones can be exposed to a broader attack surface. We categorize the attacks into two subclasses: attacks exploiting the `/proc` filesystem and those exploiting the smartphone embedded sensors.

The `/proc` is a system-level filesystem created by the kernel in Linux. It contains the system information such as interrupt and network data. Even though it is a system-level filesystem, it is readable by the user-level threads and applications. Hence, accessing the `/proc` filesystem does not require any additional permission. As a result, `/proc` has been widely employed to perform side-channel attacks. Chen *et al.* [67] proposed a UI state inference attack through which an attacker can carry out UI phishing to trick victims to make unwanted requests to edge servers by using the memory data that are publicly available in `/proc`. Diao *et al.* [68] exploited the interrupt information stored in `/proc/interrupts` to infer sensitive information of a smartphone such as pattern lock and foreground running UI. Zhou *et al.* [69] made use of various side channels such as `tcp_snd`, `tcp_rcv`, and BSSID available from an Android device's `/proc` to infer a user's sensitive information, including health condition, location, and social network identity. Xiao *et al.* [70] further strengthened this paper by developing an approach to correlate a victim's social network identity with the smartphone device used to access the social network in a more accurate and practical manner. Yet, as a matter of fact, accessing `/proc` requires an attacker to trick a victim to install a malicious app, which is the bottleneck of such attacks.

Nowadays, a smartphone is integrated with a variety of embedded sensors for handling various tasks. On the one hand, these sensors can greatly elevate the functionalities of the smartphone. On the other hand, they impose security concerns of leaking sensitive information. Asonov and Agrawal [71] and Zhuang *et al.* [72] independently showed that it is feasible to infer a user's keystroke by analyzing the acoustic sounds emitted from the physical keyboards which existed in early-stage smartphones. However, most current smartphones eliminate the physical keyboards by employing touchscreens. Nevertheless, attackers make a great effort to keep pace with this development. Zhou *et al.* [73] cracked the pattern lock of a smartphone by leveraging the acoustic signals reflected by the fingertip captured through microphones. Cai and Chen [74] showed that the tap keystrokes can be inferred using the smartphone accelerometer and gyroscope sensors. Recently, Chen *et al.* [75] proposed a novel side-channel attack leveraging the victim's eye movements from a video secretly recorded by a smartphone camera, to infer the victim's keystrokes on a mobile device. One can see that embedded sensors within a smartphone carry abundant information that can be exploited to perform inference attacks.

Based on the earlier summary, we can further categorize the side channels that can be exploited by attackers into two classes: the controllable ones that include packet streams as well as smartphone-based `/proc` filesystem and embedded sensors, to which the access can be restricted, and the uncontrollable ones that include wave signals and power consumption, which exist unconditionally due to the innate nature and are not modifiable.

2) Current Defense Solutions: The root cause of side-channel attacks is the hidden correlation, which could be very complicated and hardly identified, between the sensitive data to be protected and the publicly available side-channel information. Apparently, defenses against side-channel attacks can be performed from two directions: restricting the accesses to side-channel information and protecting the sensitive data from inference attacks. Obviously, there exists no feasible defense mechanism that can restrict the access to uncontrollable side channels, leaving sensitive data protection the only approach for such kind of attacks. In this section, we first put forward an overview on data perturbation, a well-researched technique for protecting sensitive data from inference attacks; then, we summarize the defense mechanisms that can restrict the accesses to side-channel information.

a) Data perturbation: The most well-known perturbation algorithm to protect sensitive data from inference attacks is k-anonymity, which modifies the identifier information of a piece of data before publishing its sensitive attributes, making it indistinguishable from another $k-1$ pieces of data, with these k pieces of data forming an equivalence class [76]. Machanavajjhala *et al.* [77] found that k-anonymity suffers from the homogeneity attack when the values of a sensitive attribute within an equivalence class are identical. To overcome this issue, they proposed l-diversity by ensuring each equivalence class to have at least l distinct values for each sensitive attribute [77]. However, Li *et al.* [78] pointed out that l-diversity has two main limitations, i.e., it may be difficult and unnecessary to achieve and it is insufficient to prevent attribute disclosure when the distribution of a certain value differs significantly from those of others for the same sensitive attribute. Therefore, they proposed t-closeness to overcome these two limitations by requiring the difference between the distribution of a sensitive attribute value in a class and that in the whole database is less than a threshold [78]. Nevertheless, researchers noticed that the earth mover's distance metric used in t-closeness may not be able to convincingly measure the "closeness" among values [79].

On the other hand, even though k-anonimity and its successors provide a reasonable privacy protection, they do not have sound theoretical foundations to support their privacy preservation capacity. A breakthrough was made by Dwork [80], who presented the concept of ϵ-differential privacy, which formally defines privacy preservation with a solid theoretical proof. A data randomization algorithm $\mathcal{A}$ is said to provide ϵ-differential privacy if

$$Pr[\mathcal{A}(D_1) \in S] \leq e^{\epsilon} \times Pr[\mathcal{A}(D_2) \in S] \tag{1}$$

where D_1 and D_2 are any two databases that differ by a single entry, S is the collection of the results of $\mathcal{A}$, and ϵ is a positive real number, namely, privacy budget. In the same work, Dwork *et al.* [80] also proposed the Laplace mechanism to achieve ϵ-differential privacy for numerical values by adding noises generated from a Laplace distribution. To deal with entity objects, McSherry and Talwar [81] presented the exponential mechanism to achieve differential privacy. Since differential privacy is the only privacy norm with strict mathematical proofs, it has been widely deployed to defend against side-channel information leaks. McSherry [82] implemented the Privacy Integrated Queries (PINQ) platform that provides differentially private data analysis through SQL-like programming languages. Later, Roy *et al.* [83] built Airavat, a differentially private platform for data computation over the edge servers. Xiao *et al.* [84] injected differentially private noises into `procfs` to prevent side-channel attacks on data storage. Cheu *et al.* [85] proposed a distributed differential privacy model via mixnet, which fits the decentralization feature of edge computing. Yet, the problem of differential privacy is that it may have limited protection effect if the data are correlated [86]. More generally speaking, differential privacy may not be applicable when the data have a high global sensitivity, which requires strong noises to perturb the data for privacy guarantees, thus sacrificing the utility of the data [87].

b) Restricting accesses to side channels: As mentioned earlier, an alternative approach to defend against side-channel attacks is to restrict the accesses to the side-channel information. Side-channel obfuscation on the source code level is such a scheme to defend from the software surface. Molnar *et al.* [88] proposed a mechanism to eliminate control-flow side-channel attacks from the C source code. Zhang *et al.* [89] developed a side-channel detection scheme to monitor the abnormal cache behaviors on cloud and edge servers. These two methods directly perturb the side channels to obstruct the accuracy of inference algorithms leveraged by the side-channel attacks. In recent years, as the development of TrustZone technology advances quickly, researchers invented mitigation solutions using TrustZone-empowered hardware, SGX, to prevent side-channel attacks [90], [91]. This method mainly disallows unauthorized accesses to the side channels protected in TrustZone.

3) Discussion: As mentioned earlier, the root cause of side-channel attacks is the concealed correlations between the publicly available side-channel information and the sensitive data that should be protected. Side-channel information leak is omnipresent and unavoidable, and side-channel attacks are usually highly profitable, making the attackers strongly motivated to launch such attacks. Moreover, with the advances in machine learning, especially in deep learning, successfully performing a side-channel attack is becoming much easier. On the other hand, in edge computing, most of the attacks can be done through traffic eavesdropping or malicious apps, which are easy to realize. Even though the attacks exploiting the power consumption data collected by oscilloscopes require an attacker to physically contact the edge device, as many edge devices share the same secret key (also known as master key) for a particular application, the attacker has the incentive to physically attack one of the devices at a high cost [46]. Hence, one can safely claim that all the side-channel attacks can be practically launched.

Side-channel attacks are the most difficult ones to defend against among all types of attacks considered in this paper because they can be silently and passively launched. As mentioned earlier, defense mechanisms based on data perturbation techniques, such as differential privacy, can effectively inhibit attackers from accurately inferring a user's sensitive information but they may sacrifice the data utility. Existing research showed that Apple's implementation of differential privacy results in a high privacy loss of 16 per day, while the ideal loss should not be greater than 2 [92]. Moreover, quantifying the tradeoff between privacy protection and data utility for a particular application is still generally open. On the other hand, defense solutions, such as source code-level obfuscation and hardware protection, can provide protection by directly manipulating on or restricting access to the side channels, but it is infeasible to apply them to each single piece of side-channel information since most of such information is undetectable and may vary from system to system. In addition, ironically, many of the existing solutions are vulnerable to side-channel attacks as well. The most well-known platform implementations of differential privacy, namely, PINQ and Airavat mentioned earlier, were found vulnerable to a timing attack due to the logic design flaws discovered by Haeberlen *et al.* [93]. Lee *et al.* [94] exploited the branch history channel of SGX to successfully infer the fine-grained control flow executed in SGX. Such limitations exist on both edge servers and devices.

Future research on defending against side-channel attacks in edge computing may focus on enhancing access control models to better regulate the accesses to the controllable side channels and the published data. Also, besides being used as attack resources, side channels can be employed as defense resources. The research conducted by Clark *et al.* [63] demonstrated a good example of detecting malware based on power consumption. Hence, we perceive that using side channels to enhance defenses may be a sound future research direction.

C. Malware Injection Attacks and Defense Mechanisms

The action to effectively and stealthily inject/install malware into a computing system is called malware injection attack. This type of attacks is one of the most dangerous ones since malware is a significant threat to system security and data integrity. In the traditional Internet or general-purpose computer infrastructures where strong computational power is available to support high-performance firewall or other threat protection systems, malware injection is not always feasible and possible. Nevertheless, edge devices and the low-level edge servers can barely be protected by a traditional firewall and, hence, are more vulnerable to malware injection attacks.

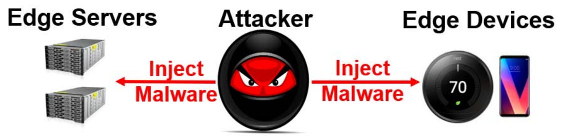


Fig. 5. ***Typical architecture of the malware injection attack.***

1) Attack Specifications: The typical architecture of the malware injection attack is straightforward. As shown in Fig. 5, the objective of the attack is to inject malware, i.e., malicious codes, into edge devices or edge servers.

We classify malware injection attacks in edge computing into two categories: server-side injections (injection attacks targeting edge servers) and device-side injections (injection attacks targeting edge devices).

a) Server-side injections: There are mainly four types of injection attacks targeting edge servers, namely, SQL injection, cross-site scripting (XSS), Cross-Site Request Forgery (CSRF) and Server-Side Request Forgery (SSRF), and Extensible Markup Language (XML) signature wrapping, which are elaborated as follows.

The SQL injection is a code injection technique that destroys the back-end databases. To construct a normal SQL query, a legitimate user is allowed to manipulate only the designated areas (e.g., name and date) to get the results from the server. However, an attacker may manage to circumvent this constraint by inputting escape characters (such as quotation marks) along with the query string. In this case, the server may mistakenly execute everything the attacker inputs after the escape characters. This vulnerability usually exists when a database management system does not filter escape characters for SQL processing. SQL injection not only is a serious threat to data confidentiality and integrity but also allows attackers to inject malicious scripts, e.g., using `SELECT ... INTO OUTFILE` command. Hence, SQL injection is one of the main methods for malware injection [95].

XSS is a client-side attack in which an attacker injects malicious codes (usually HTML/JavaScript codes) into data content, which can be accessed and executed automatically by the servers. Note that the modifier "client-side" here does not refer to the edge device side, but rather the edge server side in which an edge server works as a "client" to visit or access the services provided by other edge servers or the cloud server. Therefore, contrary to the traditional general-purpose computing systems, XSS is a type of injection attacks that happen at the edge server level in edge computing. The attack is caused by the fact that the edge servers do not filter code from data content. Even though XSS is not a novel attack and its mechanism has been well studied, it is still a serious threat to edge computing infrastructures. An XSS vulnerability was reported in the Cisco Edge Director framework that can lead to arbitrary code executions [96]. Martin and Lam [97] created an automatic model checker based on the goal-directed model checking to find XSS and SQL vulnerabilities in a large amount of codes.

CSRF is an attack in which an end user (i.e., an edge server in this case) is forced to execute unwanted actions through Web applications. SSRF is an attack in which edge servers are abused to read or alter the internal resources. The root cause of both attacks is the coarse-grained design of the verification mechanism, such as a weak identity verification method that can be easily broken. By exploiting coarse-grained verification, an attacker can spoof as a "legitimate" edge server to send a command to other edge servers without being discovered, making other edge servers to be subjected to the CSRF and SSRF attacks. Typically, CSRF and SSRF mainly target the traditional Internet infrastructures. It was found in 2016 that edge systems are also susceptible to these two attacks [98].

XML signature wrapping happens when an edge computing infrastructure acquires Simple Object Access Protocol (SOAP) as its communication protocol, which transmits the messages using the XML format. In this attack, a malicious attacker first intercepts a legitimate XML message, creates a new tag, and places a copy of the original message (which may contain verification parameters such as tokens) within the new tag (also known as the wrapper) to form a "giant" tag-value pair. Next, the attacker replaces the original values with malicious codes in the original message and combines the modified original message with the "giant" tag-value pair by placing the new pair before the regular tag-value pairs of the original message. Upon receiving this tampered message, the victim edge server would first verify the message, which could succeed since the attacker did not delete the original values (containing the still-valid verification information) but rather puts them into a new tag (wrapper). Once verification succeeds, the server would execute the malicious code injected by the attacker [99].

b) Device-side injections: Various diverse methods for injecting malware into IoT devices exist since IoT devices are highly heterogeneous on both hardware and firmware. The most common approach to remotely inject malware is to exploit the zero-day vulnerabilities that can lead to remote code execution (RCE) or command injection. One of the most infamous examples is the "IoT Reaper" virus captured in 2017, which infects millions of IoT devices through the Internet protocol and Wi-Fi by exploiting at least 30 RCE vulnerabilities existing in 9 different IoT devices ranging from the network router to IP camera [100]. In academia, Cui *et al.* [101] discovered that the HP-RFU (remote firmware update) protocol adopted by LaserJet printers allows an attacker to modify any predeployed firmware of a printer due to the lack of signature verification check. Hernandez and Buentello [102] found that the Smart Nest Thermostat lacks proper protection for firmware update, allowing an attacker to update an arbitrary firmware using a USB connection. Maskiewicz *et al.* [103] pointed out that the firmware update mechanism used by the Logitech G600 mouse is buggy and allows an attacker to infect a firmware through networking or USB. Recently, Ronen *et al.* [46] implemented an attack to remotely and contactlessly inject malicious firmware into IoT devices using the zigbee light link protocol. Note that the above-mentioned attacks are typically referred to as firmware modification attacks.

Injecting malware that has cross-accessing capability into mobile devices is not trivial since major mobile OSes, such as iOS and Android, adopt an app-isolation mechanism, i.e., the sandbox mechanism, to ensure that every app is isolated virtually on memory and no app can access other apps' resources and contents unless permitted from the kernel level [104]. Wang and Chen [105] conducted an early study to summarize how legitimate API calls, e.g., intent and scheme that are open to all mobile developers, can possibly allow an attacker to inject malicious contents into other third-party benign apps. Ren *et al.* [106] exploited the OS-level structure called Android Task Structure (ATS) to passively inject malicious UIs to benign apps. Xiao *et al.* [107] improved Ren's work by exploring active attacks exploiting ATS, making the ATS-based injection attack more feasible and powerful in practice. Even though these attacks sound powerful, they may not cause significant damages to the edge computing infrastructure as they directly exploit the Android official APIs and structures. To carry out more dangerous attacks, attackers tend to use third-party malicious libraries that are not only more powerful but also less likely to be detected. Chen *et al.* [108] found that 6.84% of official Android apps from the Google Play Store and 2.94% of iOS apps from the Apple App Store use harmful libraries such as PhaLibs that opens a backdoor for code injection. Note that all the attack mechanisms mentioned earlier require a victim to install the attacker's malicious app to begin with, since they all require the assistance from the native Android OS at some level. This limitation may diminish that the practicality for launching such attacks in the real world as a victim has to be tricked to install the malicious app in the first place. To overcome this limitation, Li *et al.* [109] made a breakthrough by discovering a serious design vulnerability in Android `WebView`, which allows an attacker to remotely inject malicious apps into a legitimate Android device through a malicious website. This method can achieve most of the attack effects (e.g., stealing other app's resources and UI hijacking) and an app-required attack can have without the need of installing a malicious app into a victim's device—it only requires the victim to open the webpage using its smartphone.

2) Current Defense Solutions: The root cause of server-side injections is the protocol-level design flaws, while the root cause of device-side injections lies in both code-level design flaws as well as the adoption of coarse-grained access control models. Accordingly, the defenses against server-side injections mainly adopt the detection filter philosophy, while those against device-side injections mainly focus on a code-level analysis for malicious behavior detection and fine-grained access control.

a) Defenses against server-side injections: Defenses against server-side injections also consider the four major types of attacks: SQL injection, XSS, CSRF/SSRF, and XML signature wrapping.

Since SQL injection attacks were discovered at the time when SQL databases were invented, defense and detection mechanisms have been studied for a fairly long time. Halfond *et al.* [110] categorized the early research into detection-focused and prevention-focused. Detection-focused techniques basically employ code checking with various schemes such as static analysis, dynamic debugging, blackbox testing, and taint-based analysis, while prevention-focused techniques target to prevent any illegal SQL queries from being executed by means such as setting up a proxy filter and employing instruction-set randomization (ISR). In the same work, Halfond *et al.* [110] also pointed out that most of the early defense mechanisms were not mature and that only two of them may have the potential to resolve practical SQL injection attacks. The first one is the lattice-based static analysis framework proposed by Huang *et al.* [111], which not only achieves defense against SQL injection on a full surface but also manages to run in real time. Yet, the major limitation of this mechanism is that it only protects the servers built on PHP, while many servers are built on other back-end languages such as Java and C#. The second mechanism was designed by Livshits and Lam [112], which is specific to Java programs, while SQL injections can happen in other back-end languages such as PHP and C#. Recently, researchers proposed improved mechanisms by comparing input queries with programmer intended queries to check inconsistencies for possible SQL injection identification or even simply remove SQL query attribute values for further analysis before a query can be executed [113], [114]. Yet, these mechanisms require a fair amount of manual effort to begin with. Therefore, researchers started to seek possible solutions from machine learning or deep learning. For examples, Jackson and Bennet [115] located SQL injection vulnerabilities in a program using NLP, and Ross *et al.* [116] evaluated multiple machine learning techniques and showed that they can achieve high accuracy for detecting SQL injection attacks.

Similar to SQL injection, the defense mechanisms against XSS have been studied for a fairly long time. Based on the survey conducted by Gupta and Gupta [117], early studies focused on ten types of defense schemes, including manually implementing hardcoded rules at the client side to inhibit XSS malicious codes from being executed and adopting ISR to turn the malicious codes into harmless ones, just to name a few. Gupta *et al.* [117] also mentioned that these early studies cannot completely resolve all types of XSS attacks. Recent research improved the early studies by adding context-aware sanitization into XSS detection to make it more robust with reduced false positive rates [118]. Rathore *et al.* [119] also leveraged learning techniques to detect XSS vulnerabilities and achieved over 97% accuracy.

Compared to SQL injection and XSS, CSRF and SSRF have a much shorter history. The number of defense mechanisms targeting CSRF is still limited. Jovanovic *et al.* [120] proposed a defense approach based on secret token, while Johnson [121] presented a CSRF defense scheme based on referer header checking. Later, Barth *et al.* [122] showed that these two mechanisms can fail if being adapted to new variations of CSRF, e.g., login CSRF; they proposed a modified version of referer header method by forcing the client side to send a `origin` header to defend against the login CSRF. Yet, this mechanism requires effort from the client side, which may increase the computational burden of edge devices. Czeskis overcame this limitation by offloading the detection task to the server side to create a lightweight protection platform. The defense against SSRF is even more limited. Fung and Lee [123] proposed a privacy-preserving defense mechanism against SSRF by embedding clients' credentials in requests. Srokosz *et al.* [124] revised the static web application firewall (WAF) approach making it able to defend against SSRF.

XML signature wrapping became popular when the SOAP protocol was introduced in the industrial world; hence, it has the shortest history among all server-side malware injection attacks. Compared to other server-side injection attacks, wrapping attacks are not difficult to defend against due to their limited attack surface and simple attack forms. Jensen *et al.* [125] proposed a schema hardening approach on the basis of W3C XML Schema for countering the wrapping attack. Gupta and Thilagam [126] developed a side-channel based detection mechanism by counting the frequency of each node in a requested service to spot suspicious wrapping attack. Kumar *et al.* [127] developed a detection method by introducing positional tokens.

b) Defenses against device-side injections: For injection attacks targeting IoT devices, as we mentioned earlier, the main threat comes from firmware modification attacks. Currently, limited research exists to investigate the corresponding defense mechanisms. To the best of our knowledge, Cui *et al.* [101] was the first to propose the defense mechanisms to mitigate firmware modification attacks. Inspired by the idea of address space layout randomization (ASLR) and ISR, they proposed the autotomic binary structure randomization (ABSR) that takes arbitrary executables or firmware as the input and outputs a variant of the original with reduction of unused codes to minimize the attack surface. They also proposed a software symbiotic method that injects intrusion detection functionality into the binary firmware of the existing IoT devices to inhibit malicious modifications. Even though the ideas of these two mechanisms are reasonable, their realizations are never easy, and Cui *et al.* [101] did not present any specific methodology on the implementation of these two mechanisms. Lee and Lee [128] proposed a blockchain-based cryptographic approach to securely update firmware for IoT devices. However, this design adopted the proof-of-work (PoW) algorithm that is not only computationally prohibitive both on hardware and on time in IoT but also lacks the ability to detect malicious codes, making this method impractical in defense. Moreover, it is unconvincing that this blockchain-based mechanism can fit the edge computing infrastructure. Weiser *et al.* [129] proposed to isolate sensitive data and codes from other nonsensitive ones using the memory protection unit (MPU) to inhibit firmware modification attacks under RISC-V. However, this approach requires that the microcontroller (MCU) of an IoT device is to be equipped with an MPU. Moreover, it only supports the RISC-V instruction set, while the Advanced RISC Machine (ARM) and the Microprocessor without Interlocked Pipelined Stages (MIPS) are more popular—in fact, ARM has approximately 95% of the IoT market share [130]. In summary, defending against malicious firmware modification attacks remains largely underexplored at the end device.

Malware injection attacks targeting mobile devices exploit the design flaws of the mobile OSes as well as the usage of malicious libraries. Schmerl *et al.* [131] and Wu *et al.* [132] independently proposed the static analysis methods to identify possible malicious uses of dangerous Android APIs. Backes *et al.* [133] developed a library detection technique that can resist the common code obfuscations and detect security vulnerabilities and malicious behaviors in Android libraries. Xiao *et al.* [107] proposed a task interference checker, TICK, to eliminate malware injection by making use of the ATS. Unfortunately, currently, no practical solution exists that can counter the remote malware injection attack exploiting *WebView* mentioned in [109] unless implementing protective schemes at the Android kernel or redesigning the Android OS.

3) Discussion: One can see that the root cause of server-side injections is the protocol-level design flaws, while those of device-side injections include code-level design flaws and the adoption of device-level coarse-grained access control. All server-side injections can be practically launched, and as a matter of fact, they have been widely exploited in industry. Even though device-side injections may not be as common as their counterparts at the server side, all the device-side injections mentioned in this section were implemented using physical IoT/mobile devices, proving that they can be launched practically in the real world.

The defense mechanisms targeting edge device malware injection/modification are quite unsatisfied. To the best of our knowledge, there exist no ideal or mature solutions to defend against the zero-day injections, firmware modification attacks, and remote WebView infections. Moreover, current solutions (e.g., code-level static analysis) provide belated actions that cannot prevent damages as they cannot be utilized as real-time weapons; additionally, they require full access to either the firmware or the source codes, which may not be available most of the time. On the other hand, the existing defenses based on weak signature verification are ineffective, and fine-grained access control for malware injection prevention sounds promising but has not been investigated. Therefore, defending malware injection attacks remain to be a grant challenge, especially at the edge device side in edge computing.

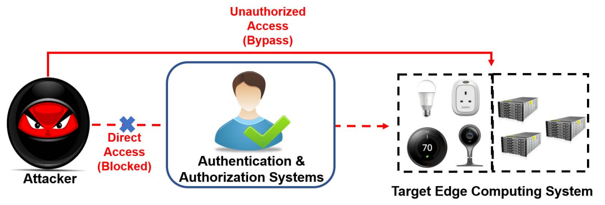

Unauthorized Access (Bypass)
Direct Access (Blocked)
Attacker
Authentication & Authorization Systems
70
Target Edge Computing System

Fig. 6. ***Typical structure of the authentication/authorization attack.***

D. Authentication and Authorization Attacks and Defense Mechanisms

Authentication is an action of verifying user identities who request certain services. Authorization is a process determining access rights and privileges of an entity, confirming that the entity behaves based on its rights without crossing boundaries. Authorization is usually proceeded with authentication for identity verification. In edge computing, authentication is generally performed between edge devices and edge servers. Under certain circumstances, it is also performed among edge devices or among edge servers in a decentralized way (so-called trigger-action platforms) [134]. Authorization in edge computing often refers to the activity when an edge server grants permissions to a certain edge device or its applications. Yet, it is also possible for devices/applications to grant permissions to other devices/applications in a trigger-action scenario, e.g., the home automation system.

1) Attack Specifications: A typical architecture of the authentication/authorization attack is shown in Fig. 6. If an attacker intends to directly access protected edge servers or edge devices, it would be blocked by the authentication system. Therefore, the attacker seeks the methods to bypass the authentication process, performing an unauthorized access.

We taxonomize the attacks into four types: dictionary attacks, attacks exploiting vulnerabilities in authentication protocols, attacks exploiting vulnerabilities in authorization protocols, and overprivileged attacks, with the first two targeting authentication protocols and the rest targeting authorization protocols. Dictionary attacks happen when an attacker employs a credential/password dictionary to crack the credential-enabled authentication system. For the attacks exploiting weaknesses in authentication and authorization protocols, we, respectively, consider WPA/WPA2 as well as SSL/TLS protocols, and the OAuth protocol, since they are the most widely adopted ones in edge computing. Overprivileged attacks occur when an app or a device is granted access rights that are stronger or more than it needs.

a) Dictionary attacks: Authentication-based attacks pose a significant threat for securing the edge computing infrastructures. The most straightforward authentication-based attack is a dictionary attack, in which an attacker possesses a dictionary containing the mostly used credentials/passwords and inputs all the possible credentials/passwords in this dictionary to the target authentication system in order to find a possible match. This type of attacks is also known as brute-force attacks, a term that is widely used in the industry. The dictionaries of commonly used passwords are extremely easy to retrieve, and plenty of public dictionaries are downloadable from open-source communities [135]. Yet, launching dictionary attacks against different protocols and different authentication mechanisms require different techniques. Lu and Cao [136] and Nam *et al.* [137] discovered that the three-party password-authenticated key exchange (S-3PAKE) protocol, which is sometimes used by Bluetooth, is vulnerable to the offline-dictionary attack right after the initial session key is established. However, employing a large dictionary can be extremely time-consuming if a single-threaded attack is used. Therefore, Nakhila *et al.* [138] proposed the parallel active dictionary attack that can be 100-fold faster than the traditional single-threaded one when attacking the WPA2-PSK Wi-Fi networks. With the development of biometric technology, adopting biometric features (e.g., fingerprints) as an authentication factor has become prevailing since people tend to believe that biometric features are more secure and tamper-proof [139]. However, this belief may not be absolutely correct. Roy *et al.* [140] proposed new schemes to create a synthetic master print that is able to match with a huge number of target fingerprints by employing the covariance matrix adaptation evolution strategy (CMA-ES), differential evolution (DE), and particle swarm optimization (PSO).

b) Exploiting weaknesses in authentication protocols: On the one hand, a dictionary attack is easy to launch; on the other hand, it has serious drawbacks such as high resource consumption and low success rate. Therefore, researchers tend to investigate more efficient attacks by discovering the design flaws of the authentication protocols. Cassola *et al.* [141] observed the weak binding vulnerability existing in the WPA enterprise authentication protocol and presented a practical and stealthy evil twin attack against WPA. Bhargavan and Leurent [142] identified a new class of transcript collision attacks targeting the TLS authentication protocol, which can result in practical impersonation and downgrade attacks. Felsch *et al.* detected a key reuse vulnerability existing in four edge/cloud vendors' authentication services, and pointed out that exploiting this vulnerability allows an attacker to arbitrarily impersonate a host or a device [143]. Vanhoef and Piessens [144], [145] identified severe design flaws in some of the OSes and platforms that allows an attacker to force nonce reuse in WPA2, then finally making it possible to replay, decrypt, and forge authentication messages.

Besides the vulnerabilities identified in WPA/WPA2 protocols, researchers also investigated the weaknesses of the authentication protocols in 4G and 5G networks. Tu *et al.* [146] developed a signaling diagnosis tool that uncovers six functional safety issues in 4G and showed that with such issues, a legitimate user can be denied 4G services. Rupprecht *et al.* [147] identified four security flaws in some implementations of 4G LTE, which allow an attacker to launch a man-in-the-middle (MITM) attack in a protected 4G LTE network and peek into sensitive unencrypted data. Hussain *et al.* [148] uncovered the security design flaws in the attach, detach, and paging procedures of the 4G LTE protocol, based on which they were able to create new attacks that can enable an adversary to spoof the location of a legitimate user without presenting correct credentials. Later, Hussain *et al.* [148] improved the work by uncovering security vulnerabilities in certain 5G implementations [149], with which an attacker can learn a victim's location and inject fabricated paging messages.

c) Exploiting weaknesses in authorization protocols: Authorization-based attacks usually exploit the design weaknesses or logic flaws existing in authorization protocols in order to achieve unauthorized access to the sensitive resources or perform privileged operations, also known as the overprivileged issue. In edge computing systems, OAuth is a widely used authorization protocol designed for multiparty authorization [150], [151]. In OAuth, three parties, i.e., a user, a service provider, and a relying party, are involved. The purpose of OAuth is to let the service provider access the user's resources (stored in the relying party) only after the user grants the access rights to the service provider. The initial version of OAuth, i.e., OAuth 1.0, which has been broken, is vulnerable to fixation attacks that happen when the service provider requests token approval from the relying party [152]. Hence, most vendors adopt OAuth 2.0 in practice. Even though OAuth 2.0 is not vulnerable at the theoretical level, some of the misimplementations can cause concerns. Chen *et al.* [152] identified that the OAuth protocol in 59.7% of the mobile applications was incorrectly implemented. Sun and Beznosov [153] analyzed the OAuth single sign-on (SSO) system for 96 relying party vendors and uncovered several critical vulnerabilities that allow an attacker to access victim's personal information without authorization.

d) Overprivilege attacks: Besides the problem of the OAuth protocol, researchers also identified overprivileged issues in typical authorization systems. Fernandes *et al.* [5] reported that the Samsung smart home platform, SmartThings, has severe overprivileged issues, allowing an attacker to develop malicious SmartApps that can launch overprivileged attacks such as changing door pin and falsely turning on a fire alarm. Jia *et al.* [154]

developed a graph-based algorithm to automatically excavate the overprivileged weaknesses in a smart home system. Based on these weaknesses, they created several attacks to control victim's smart home devices without authorization. Felt *et al.* [155] found that about one-third of the Android apps have the overprivileged problem. Ho *et al.* [156] conducted security analysis on the current off-the-shelf smart lock products and found that all of them are vulnerable to at least one of the following uncovered attacks: allowing an attacker to evade a device revocation mechanism or a smart lock app/protocol to undesirably unlock a user's door.

2) Current Defense Solutions: The root cause of dictionary attacks is the adoption of weak credentials in authentication protocols, while the root cause of the other three types of attacks is the protocol-level design flaws or the implementation-level flaws. Accordingly, the defenses against dictionary attacks mainly focus on adding a stronger authentication layer or hardening the password verification processes, and the defenses against the other three types of attacks mainly adopt the philosophy of patching/strengthening the current protocols or conducting code-level analyses. Authentication is the security entry point of a system. Most attackers take down edge devices or edge servers beginning with compromising the authentication systems at the first place. Therefore, securing authentication in an edge computing system is critical.

a) Defending against dictionary attacks: In Section III-D1a, we demonstrate a fair amount of work exploiting dictionary attacks to break possible weak credentials in edge computing. Intuitively, one may perceive that simply mandating complicated passwords for authentication may resolve the problem; however, this may not be feasible for at least three reasons. First, due to the limited computation power issue in edge infrastructures, adopting complicated passwords may increase the computation overheads. Second, unlike the traditional Internet applications such as social networks, edge computing infrastructures have way more subscribers, i.e., edge devices. Using complicated passwords increases the storage burden. Third, storing credentials in IoT devices is not secure since they are fragile on security and are more vulnerable to password leakage. Therefore, researchers have been seeking alternatives to defend against dictionary attacks. Many existing studies considered raising the cost of dictionary attacks by adding one more layer of authentication, which is known as two-factor authentication. Pinkas and Sander proposed the first two-factor authentication technique by adding a challenge that can be easily answered by human beings but infeasible to be answered by automated programs from dictionary attacks [157]. The most well-known two-factor authentication mechanisms use various features as the second authenticator, e.g., fingerprints [158], face authentication [159], authentication code via SMS messages [160], graphic texts [161], or even ambient sound [162]. However, all these two-factor authentication schemes more or less require human interactions such as inputting authentication codes, which are hardly applicable to the edge computing systems where automated edge devices operate on. On the other hand, unfortunately, all the two-factor authentication techniques mentioned earlier were proven to be insecure practically. Mulliner *et al.* [163] and Wang *et al.* [164], respectively, showed that SMS-based authentication is vulnerable to mobile Trojan and brute-force attacks. Joshi *et al* [165] proposed multiple attack vectors targeting the current fingerprint authentication systems. Zhou *et al.* [166] showed that by illuminating an attacker using infrared, the attacker can impersonate a target victim against a face recognition system with over 70% of success rate. Wang *et al.* [167] demonstrated that using a convolutional neural network (CNN) can defeat graphic-text authentication. Shrestha *et al.* [168] pointed out that employing ambient sound as an authentication factor can be easily cracked with almost zero effort. These results indicate that exploring human-less more secure two-factor authentication mechanisms is desperately needed. There also exists research that focuses on inhibiting dictionary attacks by revising the current authentication protocols. Boneh *et al.* [169] proposed using the Balloon password-hashing algorithm based on memory-hard functions to raise the cost of offline-dictionary attacks. Nevertheless, this defense mechanism significantly sacrifices authentication efficiency. Jarecki *et al.* [170] introduced a device-enhanced password-authenticated key exchange (DEPAKE) protocol to mitigate the online and offline dictionary attacks without the need of public key infrastructure (PKI) settings. This scheme can effectively inhibit offline dictionary attacks but has a limited impact on online dictionary attacks.

b) Hardening authentication protocols: To defend against attacks exploiting the vulnerabilities in authentication protocols, researchers chose to either enhance the security of the communication protocols or secure the cryptographic implementations. Liu [171] proposed to use active jammer and wireless packet injection to inhibit the brute-force attack for decrypting the WPA traffics. Noh *et al.* [172] revised the original key exchange process in the WPA/WPA2 protocol by adopting the public key cryptography which can reduce the threat of several vulnerabilities including the evil twin. Nevertheless, these solutions cannot prevent the key reuse problem in WPA/WPA2. On the other hand, to secure the cryptographic implementations, Sivakorn *et al.* [173] proposed a blackbox verification mechanism to prevent possible hostname impersonations; Bhargavan *et al.* [174] leveraged symbolic execution to detect whether a TLS 1.3 implementation is vulnerable to various well-known attacks existing in TLS 1.2 such as Logjam or the Triple Handshake. These mechanisms are offline ones that cannot achieve real-time protection but can minimize the attack surfaces against the TLS implementations.

To harden the 4G and 5G network protocols, Tu *et al.* [146] proposed a three-step solution by: 1) adding a slim layer for signaling transferring; 2) decoupling domains in the 4G radio resource control (RRC) layer; and 3) coordinating similar functions at different systems. By doing so, one can ensure that a legitimate user would not be denied services in a 4G network. Duan and Wang [175] proposed a theoretical protocol for fast authentication in 5G heterogeneous networks (HetNets) using SDN. Zhao *et al* [176] developed a lightweight scalable secure cross-layer authentication architecture for 5G by leveraging RF fingerprints as a piece of evidence to defend against possible impersonation-related attacks. Ni *et al.* [177] proposed a cryptographically secure service-oriented authentication protocol for 5G-Enabled IoT using network slicing. Yet, even though these methods can strengthen the 4G and 5G network protocols from different perspectives, securing them practically at the implementation level might still be a long-haul journey.

c) Hardening authorization protocols: Authorization in an edge computing infrastructure is equally important compared to the authentication system. Without a properly designed authorization mechanism, an attacker can exploit the weaknesses in the authorization system as well. As mentioned earlier, many implementations of the OAuth protocol are problematic, even though OAuth 2.0 is theoretically secure. Yang *et al.* [178] proposed a static code analysis method to check and fix the OAuth implementation vulnerabilities based on three OAuth service providers: Google, Facebook, and Sina. Shehab and Mohsen [179] developed an application-based OAuth Manager framework to prevent the misuses of the current OAuth APIs. Cirani *et al.* [180] implemented an OAuth protocol specifically targeting HTTP/CoAP services to provide authorization for IoT-based applications.

d) Defending against overprivileged attacks: Overprivileged issues mainly exist in IoT systems and mobile devices. Tian *et al.* [181] proposed an NLP-based method to check the inconsistencies between an IoT App's implementation and its description in order to identify overprivileged exploits. Celik *et al.* [182] developed a taint-based analysis for tracking and preventing the sensitive information leakage due to overprivileged designs of IoT apps. Bastys *et al.* [183] proposed an information tracking technique to monitor possible information leak from overprivileged apps. Celik *et al.* [184] successfully identified 20 flawed apps among the 35 SmartApps on the SmartThings platform using IoTGuard, a model-checking-based solution, to automatically identify overprivileged apps. These mechanisms need the authorizer to be able to access the source code of an IoT app, which may not be available most of the time. Jia *et al.* [185] proposed a mechanism to prevent overprivileged attacks by comparing the "contexts" of behaviors with the ones in the past to identify possible suspicious inconsistencies. Similarly, Schuster *et al.* [185] developed a technique called environmental situation oracles (ESOs) to enforce IoT access control through situational environments [186], which are almost identical to the concepts of "contexts." This technique requires not only a fine-grained definition on contexts but also a long time for the legitimate contexts to be built, making the mechanism infeasible to be deployed in practice.

Strengthening the current permission models adopted by mobile OSes is a common approach to defending against the overprivileged attacks targeting smartphone devices. Wang and Chen [187] developed a semantic permission generator for Android, which interprets an app's description and grants the permissions needed by the app based on the interpretation. This semantic-based method assumes that the app developer is honest when writing the app's description, ignoring the possibility that the developer can be a malicious attacker as well. To overcome this issue, Fernandes *et al.* [188] developed a system-level sandbox to place the codes that are in charge of requesting sensitive permissions. These codes would then be isolated for further analysis to ensure that no overprivileged permissions can be granted mistakenly. Yet, this method requires not only the user to root the device for installing the sandbox but also a fair amount of manual efforts. To mitigate these limitations, Sikder *et al.* [189] proposed 6thSense, a mechanism to inhibit the permission abuse toward Android sensors from overprivileged apps by utilizing three different machine learning models, i.e., Markov chain, Naive Bayes, and logistic model tree (LMT). Aafer *et al.* [190] proposed a technique to effectively locate the overprivileged components of an Android app based on a graph abstraction algorithm and the logical reasoning algorithm.

3) Discussion: Obviously, the root cause of dictionary attacks is the adoption of weak credentials, while the root cause of the other three attacks is the protocol-level design flaws or the implementation-level flaws. Launching dictionary attacks in edge computing systems is relatively easy, as an attacker can build a credential dictionary with little or a reasonable amount of effort. Nevertheless, defending against dictionary attacks might be hard, as attackers may have their own channels to build their own dictionaries that may contain abundant information to crack the authentication system. Moreover, two-factor defense mechanisms were proven to be ineffective as all were broken practically. On the other hand, attacks exploiting the weaknesses of authentication and authorization protocols need more effort to conduct, as the attackers have to identify the protocol vulnerabilities and compromise the edge servers. Nevertheless, cracking an authentication/authorization protocol implies that the attackers can break into the system and then launch various other attacks. Overprivileged attacks are common in IoT and mobile devices, and many real-world overprivileged apps have been identified [155], [181].

Yet, the corresponding defenses are generally *ad hoc* and ineffective.

E. Comparison Study on Security Issues in Edge Computing and Cloud Computing

Having discussed the attacks and defense solutions for edge computing, we now compare the security issues between cloud computing and edge computing. Cloud computing is a practice of utilizing centralized remote servers and data centers in the Internet for data storage, processing, and analysis. The end devices in a cloud computing system are typically fully fledged computers that connect to the cloud servers mainly through the wired Internet. Note that some "so-called" cloud systems, such as SmartThings whose end units are IoT devices, usually delegate most of the services to local hubs that handle the IoT services at different regions; hence, they are not strictly cloud computing platforms [5]—such systems actually follow the edge computing paradigm. Edge computing, on the other hand, adopts a decentralized hierarchical design that may involve low-profile computers and devices as edge servers [191]. The end units in an edge computing system are typically IoT and mobile devices that are much more resource-constrained compared with fully fledged computers, and they connect to the edge servers mainly through wireless protocols such as Wi-Fi and 4G/5G. Due to this architectural difference, edge computing can provide location-aware, bandwidth-sufficient, real-time, privacy-enhanced, and low-cost services compared to cloud computing, as we mentioned in Section I. Therefore, even though edge computing and cloud computing are similar with respect to the offered services and functionalities, the scopes of attack measures are largely different. We summarize the major differences between cloud computing and edge computing with respect to security attacks in the following paragraphs.

1) DDoS Attacks: In flooding-based DDoS attacks, attackers tend to exploit the vulnerabilities of network protocols of the target system. Clouding computing typically employs heavyweight network protocols such as HTTP/HTTPS, FTP, and SMTP as cloud servers have more computational resources for strong security protocols, while edge computing typically adopts lightweight network protocols such as CoAP, MQTT, and UDP due to the resource constraints of edge servers. Such a difference makes the attacking effort significantly larger in a cloud computing system. On the other hand, since compromising fully fledged computers are becoming more and more difficult as the firewalls running on them are getting more and more powerful, DDoS attacks may no longer be that effective in cloud computing. However, as most of the edge devices are low-end computing units with limited computational power, it is easier for an attacker to first compromise a number of edge devices and then exploit them to launch flooding-based DDoS attacks (such as Mirai [7]). Hence, flooding-based DDoS is still practically effective in edge computing systems. Nevertheless, both edge computing and cloud computing suffer from zero-day DDoS attacks as code-level flaws can happen regardless of the difference of the computing platforms. According to an official report published by CVE, in the year of 2018, cloud computing infrastructures suffer from more zero-day vulnerabilities than edge computing infrastructures do, indicating that cloud computing may be more vulnerable to zero-day DDoS attacks [192].

2) Side-Channel Attacks: Attackers have a wider attack surface to launch side-channel attacks in edge computing as edge devices are typically connected wirelessly and they are more approachable to attackers. More specifically, attackers in cloud computing often choose to extract valuable information through eavesdropping [193] or carefully crafted queries in side-channel attacks as the cloud devices are generally protected by relatively powerful firewalls [194]. Therefore, cloud computing is typically vulnerable to the side-channel attacks that exploit packet-based communication channels, and attacks exploiting wave signals are generally unfeasible in cloud computing since wired connections instead of the wireless ones are adopted. In edge computing, attackers have more side channels to manipulate. They can easily access diverse side-channel information by either compromising or staying close to an edge device, launching various device-side side-channel attacks such as those exploiting the wireless communication channels, power consumption, and smartphone-based channels.

3) Malware Injection Attacks: Generally, in malware injection attacks targeting cloud computing, attackers tend to first launch server-side injections to take down a cloud server and then infect the client devices via the compromised cloud server, while in edge computing, attackers tend to first launch device-side injections to take down a number of edge devices and then abuse them to take down an edge server [193]. The reason behind such a phenomenon lies in that client devices in cloud computing are usually not interconnected, and thus, taking down a few of them may have limited influential impact; on the other hand, edge devices in edge computing are typically interconnected, and thus, taking down a few can have a much more significant impact if the attacker spreads its attacks among these interconnected devices. In addition, since compromising a cloud device is much more difficult than compromising an edge device, malware injection attacks are more specific and popular in edge computing systems. This is in accord with the practice in which the majority of device-side injection attacks target IoT and mobile devices.

4) Authentication and Authorization Attacks: While both edge computing and cloud computing are vulnerable to dictionary attacks, edge devices have a higher chance of employing weak credentials than those in cloud computing [195], making dictionary attacks more popular and practical in edge computing. Also, edge computing is vulnerable to attacks exploiting weaknesses in wireless-based authentication protocols such as WPA/WPA2, Bluetooth, and 4G/5G, while cloud computing is vulnerable to attacks exploiting weaknesses in those such as SSL/TLS that can be applied to wired protocols. On the other hand, even though both edge computing and cloud computing are vulnerable to attacks exploiting weaknesses in authorization protocols, edge computing is more susceptible compared with cloud computing because cloud computing has relatively simple authorization scenarios due to the centralized design which involves less number of communication parties. Nevertheless, edge servers are deployed in a decentralized way, involving more communication parties and complicating the whole authorization process. In addition, overprivileged attacks mainly occur in IoT and mobile devices. They are typically specific to edge computing systems since such systems have much more complex access control scenarios than general-purpose computing systems do.

IV. ROOT CAUSES, GRAND CHALLENGES, AND FUTURE RESEARCH

In this section, we first summarize the root causes of security threats and outline the remaining grand challenges; we then propose a few potential future research directions toward strengthening the edge computing security.

A. Root Causes of Security Issues

As mentioned in Section III, current edge computing infrastructures suffer from serious cyberattacks, which may result in huge financial loss. The leading root causes of these security threats lie in the unavoidable flaws or vulnerabilities in protocol and code designs as well as their implementations, the concealed correlations between the public and the protected private/secure data, and the lack of fine-grained access control, which are elaborated as follows.

1) Protocol-Level Design Flaws: Protocol is the fundamental support for basic functions such as communications and data processing in edge computing from the theoretical perspective. Therefore, designing secure protocols is critical to maintain a healthy edge computing ecology. Unfortunately, as shown in Section III, many of the current protocols adopted by edge computing systems have design flaws because their designers mainly focus on utility and user experience, treating security as something unimportant if not unnecessary. By exploiting the design flaws, an attacker can achieve attack goals ranging from simply shutting down a normal service (e.g., DDoS) to fully controlling an edge device or server (e.g., malware injection attacks).

2) Implementation-Level Flaws: Implementation is the process of practically realizing the edge computing functionalities. Even if a protocol may be proven secure mathematically, it does not necessarily mean that its implementation is secure. Logic flaws in an implementation can neutralize the security strength of a protocol which has been proven strictly secure. There are two main reasons leading to the implementation-level flaws: 1) developers may misunderstand the foundations of the protocol and 2) migrating a protocol from other platforms to the edge computing platform may cause adaptivity inconsistencies. Implementation flaws can be equally serious as the protocol-level design flaws. By exploiting implementation flaws, an attacker can bypass the security features provided by the protocol (e.g., authentication attacks).

3) Code-Level Vulnerabilities: Code is the basic unit of a program, which defines the exact execution flow a processor should follow. As mentioned in Section III, many attacks are, in fact, the results of the code-level vulnerabilities. Note that implementation-level flaws and code-level vulnerabilities are two different concepts, with the former mainly referring to the logic flaws in a practical realization and the latter mainly referring to the system bugs that can cause memory failures/corruptions. Such vulnerabilities include stack/heap overflow, use-after-free dangling pointer, format string, and so on [196]. The presence of such vulnerabilities lies in the thoughtless programming process when coding for millions of lines of codes. By exploiting these vulnerabilities, an attacker can achieve attack goals ranging from simply shutting down a normal service to fully controlling an edge device or server (e.g., malware injection attacks).

4) Data Correlations: In an edge computing system, tons of data would be generated from the edge infrastructures constantly, with some of them being sensitive and placed under strict protections, while others are less sensitive and exposed to the public without protection. However, there may exist hidden correlations between the sensitive data and the insensitive data, which may not be straightforward. Nevertheless, by exploiting these correlations and leveraging various attack models, an attacker can infer the sensitive data (e.g., side-channel attacks) or even tamper them (e.g., bad-data injection attacks) based on the insensitive data.

5) Lacking Fine-Grained Access Controls: Access control is one of the most important security measures, through which an entity is permitted to access only the least resources it needs. Such a measure has a good reputation in protecting traditional general-purpose computing systems for a long time, but it cannot be directly adapted to edge computing due to the more complex and fine-grained permission scenarios. Many current edge computing systems implement only coarse-grained access control mechanisms or even do not employ any access control mechanism. Lacking a fine-grained access control specifically designed for edge computing applications may significantly lower the bar of launching an attack (e.g., authorization attacks and man-in-the-middle attacks).

B. Status Quo and Grand Challenges

After summarizing the major root causes of security threats, we then present the status quo and grand challenges in securing an edge computing system that a researcher and a practitioner need to be aware of.

1) Lacking Consideration of Security-by-Design: The primary goal of edge computing is to provide a more efficient and lightweight computing platform for emerging applications such as IoT and smart cities. Therefore, designers tend to focus more on performance rather than security when designing an application-specific edge computing architecture. Such an insufficient consideration of security-by-design directly exposes the edge computing infrastructures to broader attack surfaces.

2) Nonmigratability of Security Frameworks: The security frameworks for traditional general-purpose computing systems have been extensively studied for a fairly long time and they are regarded as having the ability to provide strong security guarantees in defending against various attacks. Yet, these security frameworks cannot be directly migrated to edge computing systems due to several unresolvable gaps such as contrasting computation power, diverse OSes and software, different network topologies, and disparate protocols. Moreover, the security frameworks designed for one edge computing application may not be directly migratable to another scenario due to multiple reasons such as heterogeneity of edge devices and communication protocols.

3) Fragmented and Coarse-Grained Access Control: Current access control models for edge computing are fragmented and coarse-grained. They are fragmented because different edge computing scenarios may adopt different access control models that may be designed in totally different manners with respect to permission segregating, granting, and accessing. This situation inhibits the development of a unified and manageable access control framework for various edge computing systems. Current access control models are also coarse-grained as fine-grained permissions that are specific to edge computing are complex and underexplored.

4) Isolated and Passive Defense Mechanisms: Current defense mechanisms for edge computing are isolated and passive. They are isolated because each defense mechanism shown in Section III may only be effective in countering one or a few attacks but ineffective for the majority of other attacks. They are passive because most of the defense solutions are executed based on preset rules and do not have the ability to conduct autonomous and active defense actions. These two weaknesses result in a rigid and fixed defense surface, making most current defense solutions adopt a philosophy of "detect then patch," which may only be effective after the occurrence of the attacks being detected.

C. Future Research Directions

Based on the summaries on root causes, status quo, and grand challenges of securing edge computing systems, one can conclude that research on edge computing security is far from satisfied and future research directions lie in overcoming the grand challenges as well as the existing vulnerabilities and weaknesses. On the one hand, stronger defense solutions, especially preventive mechanisms, to mitigate individual attacks are definitely needed; on the other hand, new architectures that can incorporate security mechanisms to protect the whole system in a more unified way are worthy of exploration. Most importantly, the philosophy of security-by-design should be broadly adopted and always retrospected. In the following, we outline a basic idea that intends to secure edge computing systems with a unified framework and present the future directions along this line of research. The framework contains three layers: an outer fine-grained access control layer, a middle-security function layer, and an inner hardware-isolated OS layer.

The outer layer focuses on fine-grained access control, which serves as a "door" to block the penetrations from outside attackers. We perceive a design to employ a permission model that can at least incorporate the information of the following five components: who, when, where, what, and how. If properly designed and strictly enforced, such a fine-grained access control mechanism may have the potential to mitigate attacks raised by protocol-level design flaws, implementation-level flaws, and weak access controls, which may result in flooding-based DDoS, attacks exploiting controllable side channels, malware injection attacks, and authentication and authorization attacks.

The middle layer intends to implement full-fledged strong security mechanisms. We perceive to adopt SDN and network function virtualization (NFV) at the edge server level, in which SDN is adopted to filter malicious traffics on a per-packet basis, while the NFV adopts more advanced algorithms such as deep learning to detect malicious behaviors in an autonomous and self-evolving manner. The SDN and NFV empowered edge servers may have the potential to inhibit packet-based attacks such as DDoS, attacks raised from correlated data (requiring learning-based detection), and weak access controls (that may result in attacks such as malware injections).

The inner layer targets the unavoidable code-level vulnerabilities. We perceive that designing a hardware-isolation empowered OS at the edge device layer should be a viable approach. Such an OS isolates the OS kernel and all the sensitive device-level data into different secure spaces using hardware-isolation so that they can be immune to software bugs. This design may have the potential to mitigate security issues caused by code-level vulnerabilities such as zero-day DDoS attacks and zero-day malware injection attacks.

The major purpose of presenting the three-layer security framework mentioned earlier is to "throwing bricks and getting jades." The research and development on edge computing security are still in their infancy stages. Driven by emerging applications as well as advances in modern cryptography, innovative designs and implementations to secure edge computing systems will thrive in the foreseeable future.

V. CONCLUSION

In this paper, we provided a comprehensive survey on the most influential and basic attacks as well as the corresponding defense mechanisms that can be practically applied in edge computing systems. We also analyzed the root causes of the attacks, summarized the status quo and grand challenges in securing edge computing systems, and proposed our future research. ■

Acknowledgments

This work was completed when Y. Xiao, Y. Jia, C. Liu, and X. Cheng visited the School of Computer Science and Technology, Shandong University, as Visiting Scholars.

REFERENCES

[1] (2017). *Edge Computing Market Size Forecast in the United States From 2017 to 2025, by Segment (in Million U.S. Dollars)*. [Online]. Available: https://www.statista.com/statistics/909308/united-states-edge-computing-market-size-segment/

[2] (2019). *Number of IoT Devices in Use Worldwide From 2009 to 2020 (in Billion Units)*. [Online]. Available: https://www-statista-com.proxygw.wrlc.org/statistics/764026/number-of-iot-devices-in-use-worldwide/

[3] D. R. Butenhof, *Programming With POSIX threads*. Reading, MA, USA: Addison-Wesley, 1997.

[4] B. Gallmeister, *POSIX.4 Programmers Guide: Programming for the Real World*. Sebastopol, CA, USA: O'Reilly Media, Inc., 1995.

[5] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 636–654.

[6] M. Antonakakis *et al.*, "Understanding the Mirai botnet," in *Proc. 26th USENIX Secur. Symp.* Vancouver, BC, Canada: USENIX Association, 2017, pp. 1093–1110.

[7] (2017). *Financial Impact of Mirai DDoS Attack on Dyn Revealed in New Data*. [Online]. Available: https://www.corero.com/blog/797-financial-impact-of-mirai-ddos-attack-on-dyn-revealed-in-new-data.html

[8] (2018). *Botnet: No Jailtime for Mirai-Creators*. [Online]. Available: https://www.gdatasoftware.com/blog/2018/09/31124-botnet-no-jailtime-for-mirai-creators

[9] (2017). *Most Commonly Encountered Types of Malware Used in Industrial Cyber Attacks Worldwide in 2017*. [Online]. Available: https://www.statista.com/statistics/271037/distribution-of-most-common-malware-file-types/

[10] (2019). *2017 was 'Worst Year Ever' in Data Breaches and Cyberattacks, Thanks to Ransomware*. [Online]. Available: https://www.techrepublic.com/article/2017-was-worst-year-ever-in-data-breaches-and-cyberattacks-thanks-to-ransomware/

[11] M. A. Mazidi, R. D. McKinlay, D. Causey, and P. Microcontroller, *Embedded Systems*. Hoboken, NJ, USA: Pearson, 2008.

[12] R. Barry, *FreeRTOS Reference Manual: API Functions and Configuration Options*. Bristol, U.K.: Real Time Engineers Limited, 2009.

[13] B. Xiong, "RT-thread programming guide release 0.3.3," Tech. Rep.

[14] M. Mazzillo *et al.*, "Silicon photomultiplier technology at STMicroelectronics," *IEEE Trans. Nucl. Sci.*, vol. 56, no. 4, pp. 2434–2442, Aug. 2009.

[15] L. Gwennap, "Qualcomm tips Cortex-a57 plans: Snapdragon 810 combines eight 64-bit CPUs, LTE baseband," Microprocessor Rep., Linley Group, Mountain View, CA, USA, Tech. Rep., 2014.

[16] Y. Yu, A. Miyaji, M. H. Au, and W. Susilo, "Cloud computing security and privacy: Standards and regulations," *Comput. Standards Interfaces*, vol. 54, pp. 1–2, Nov. 2017.

[17] S. Basu *et al.*, "Cloud computing security challenges & solutions—A survey," in *Proc. IEEE 8th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2018, pp. 347–356.

[18] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other Botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[19] L. Xiaoming, V. Sejdini, and H. Chowdhury, "Denial of service (DoS) attack with UDP flood," School Comput. Sci., Univ. Windsor, Windsor, ON, Canada, 2010.

[20] J. Udhayan and R. Anitha, "Demystifying and rate limiting ICMP hosted DoS/DDoS flooding attacks with attack productivity analysis," in *Proc. IEEE Int. Adv. Comput. Conf.*, Mar. 2009, pp. 558–564.

[21] M. Bogdanoski, T. Shuminoski, and A. Risteski, "Analysis of the SYN flood DoS attack," *Int. J. Comput. Netw. Inf. Secur.*, vol. 5, pp. 1–11, Jun. 2013.

[22] K. Elleithy, D. Blagovic, W. Cheng, and P. Sideleau, "Denial of service attack techniques: Analysis, implementation and comparison," *J. Syst., Inform.*, vol. 3, pp. 66–71, Jan. 2006.

[23] A. Dhanapal and P. Nithyanandam, "An effective mechanism to regenerate HTTP flooding DDoS attack using real time data set," in *Proc. Int. Conf. Intell. Comput., Instrum. Control Technol. (ICICICT)*, 2017, pp. 570–575.

[24] F. Wiens and A. Zitzmann, "Predation on a wild slow Loris (*Nycticebus coucang*) by a reticulated Python (*Python reticulatus*)," *Folia Primatol.*, vol. 70, no. 6, pp. 362–364, 1999.

[25] (2010). *CVE-2010-3972 Detail*. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2010-3972

[26] Y.-H. Hu, H. Choi, and H.-A. Choi, "Packet filtering to defend flooding-based DDoS attacks [Internet denial-of-service attacks]," in *Proc. IEEE/Sarnoff Symp. Adv. Wired Wireless Commun.*, Apr. 2004, pp. 39–42.

[27] J. Zhang, Z. Qin, L. Ou, P. Jiang, J. Liu, and A. X. Liu, "An advanced entropy-based DDOS detection scheme," in *Proc. Int. Conf. Inf. Netw. Autom. (ICINA)*, vol. 2, Oct. 2010, pp. V2-67–V2-71.

[28] A. Yaar, A. Perrig, and D. Song, "Pi: A path identification mechanism to defend against DDoS attacks," in *Proc. Security Privacy*, May 2003, pp. 93–107.

[29] H. Luo, Y. Lin, H. Zhang, and M. Zukerman, "Preventing DDoS attacks by identifier/locator separation," *IEEE Netw.*, vol. 27, no. 6, pp. 60–65, Nov./Dec. 2013.

[30] W. J. Buchanan, F. Flandrin, R. Macfarlane, and J. Graves, "A methodology to evaluate rate-based intrusion prevention system against distributed denial-of-service (DDoS)," *Cyberforensics*, 2011.

[31] R. Xu, W.-I. Ma, and W. Zheng, "Defending against UDP flooding by negative selection algorithm based on eigenvalue sets," in *Proc. 5th Int. Conf. Inf. Assurance Secur.*, vol. 2, Aug. 2009, pp. 342–345.

[32] J. Zhang, Z. Qin, L. Ou, P. Jiang, J. Liu, and A. X. Liu, "An advanced entropy-based DDOS detection scheme," in *Proc. Int. Conf. Inf., Netw. Autom. (ICINA)*, vol. 2, Oct. 2010, pp. V2-67–V2-71.

[33] S. Yu, W. Zhou, R. Doss, and W. Jia, "Traceback of DDoS attacks using entropy variations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 3, pp. 412–425, Mar. 2011.

[34] K. Kumar, R. C. Joshi, and K. Singh, "A distributed approach using entropy to detect DDoS attacks in ISP domain," in *Proc. Int. Conf. Signal Process., Commun. Netw.*, Feb. 2007, pp. 331–337.

[35] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Usilng machine learning technliques to identify botnet traffic," in *Proc. 31st IEEE Conf. Local Comput. Netw.*, Nov. 2006, pp. 967–974.

[36] M. Zolotukhin, T. Hämäläinen, T. Kokkonen, and J. Siltanen, "Increasing Web service availability by detecting application-layer DDoS attacks in encrypted traffic," in *Proc. 23rd Int. Conf. Telecommun. (ICT)*, May 2016, pp. 1–6.

[37] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based DDoS detection system in software-defined networking (SDN)," 2016, *arXiv:1611.07400*. [Online]. Available: https://arxiv.org/abs/1611.07400

[38] F. Qin, S. Lu, and Y. Zhou, "SafeMem: Exploiting ECC-memory for detecting memory leaks and memory corruption during production runs," in *Proc. 11th Int. Symp. High-Perform. Comput. Archit.*, Feb. 2005, pp. 291–302.

[39] S. Chen, J. Xu, N. Nakka, Z. Kalbarczyk, and R. K. Iyer, "Defeating memory corruption attacks via pointer taintedness detection," in *Proc. Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2005, pp. 378–387.

[40] Y. Shoshitaishvili, R. Wang, C. Hauser, C. Kruegel, and G. Vigna, "Firmalice—Automatic detection of authentication bypass vulnerabilities in binary firmware," in *Proc. NDSS*, 2015, pp. 1–15.

[41] S. V. Gupta and P. Shenoy, "System and methods for run time detection and correction of memory corruption," U.S. Patent 8 510 596, Aug. 13, 2013.

[42] M. Muench, J. Stijohann, F. Kargl, A. Francillon, and D. Balzarotti, "What you corrupt is not what you crash: Challenges in fuzzing embedded devices," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, CA, USA, Feb. 2018, pp. 1–15.

[43] Z. L. Chua, S. Shen, P. Saxena, and Z. Liang, "Neural nets can learn function type signatures from binaries," in *Proc. 26th USENIX Secur. Symp. (USENIX Security)*. Vancouver, BC, Canada: USENIX Association, 2017, pp. 99–116. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/chua

[44] W. Song, H. Yin, C. Liu, and D. Song, "DeepMem: Learning graph neural network models for fast and robust memory forensic analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2018, pp. 606–618. [Online]. Available:

http://doi.acm.org/10.1145/3243734.3243813

[45] F. Zuo, X. Li, Z. Zhang, P. Young, L. Luo, and Q. Zeng, "Neural machine translation inspired binary code similarity comparison beyond function pairs," in *Proc. NDSS*, 2019, pp. 1–15.

[46] E. Ronen, C. O. Flynn, A. Shamir, and A.-O. Weingarten, "IoT Goes nuclear: Creating a ZigBee Chain reaction," in *Proc. IEEE Symp. Secur. Privacy*, May 2017, pp. 195–212.

[47] J. Chen *et al.*, "IoTFuzzer: Discovering memory corruptions in IoT through app-based fuzzing," in *Proc. NDSS*, 2018, pp. 1–15.

[48] T. Frassetto, P. Jauernig, C. Liebchen, and A.-R. Sadeghi, "IMIX: In-process memory isolation extension," in *Proc. 27th USENIX Secur. Symp. (USENIX Security)*. Baltimore, MD, USA: USENIX Association, 2018, pp. 83–97.

[49] S. Shirali-Shahreza and Y. Ganjali, "Protecting home user devices with an SDN-based firewall," *IEEE Trans. Consum. Electron.*, vol. 64, no. 1, pp. 92–100, Feb. 2018.

[50] C. Dietz *et al.*, "IoT-botnet detection and isolation by access routers," in *Proc. 9th Int. Conf. Netw. Future (NOF)*, Nov. 2018, pp. 88–95.

[51] H. Li, Y. He, L. Sun, X. Cheng, and J. Yu, "Side-channel information leakage of encrypted video stream in video surveillance systems," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[52] X. Ji, Y. Cheng, W. Xu, and X. Zhou, "User presence inference via encrypted traffic of wireless camera in smart homes," *Secur. Commun. Netw.*, vol. 2018, Sep. 2018, Art. no. 3980371.

[53] N. Apthorpe, D. Reisman, and N. Feamster, "A smart home is no castle: Privacy vulnerabilities of encrypted IoT traffic," in *Proc. NDSS*, 2017, pp. 1–6.

[54] W. Chen and Z. Qian, "Off-path TCP exploit: How wireless routers can jeopardize your secrets," in *Proc. 27th USENIX Secur. Symp. (USENIX Security)*. Baltimore, MD, USA: USENIX Association, 2018, pp. 1581–1598.

[55] M. Enev, S. Gupta, T. Kohno, and S. N. Patel, "Televisions, video privacy, and powerline electromagnetic interference," in *Proc. 18th ACM Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2011, pp. 537–550. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046770

[56] J. Selvaraj, G. Y. Dayanıklı, N. P. Gaunkar, D. Ware, R. M. Gerdes, and M. Mina, "Electromagnetic induction attacks against embedded systems," in *Proc. Asia Conf. Comput. Commun. Secur.*, 2018, pp. 499–510.

[57] M. Li *et al.*, "When CSI meets public WiFi: Inferring your mobile phone password via WiFi signals," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2016, pp. 1068–1079. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978397

[58] I. Martinovic, D. Davies, M. Frank, D. Perito, T. Ros, and D. Song, "On the feasibility of side-channel attacks with brain-computer interfaces," Presented as the 21st USENIX Secur. Symp. (USENIX Security). Bellevue, WA, USA: USENIX, 2012, pp. 143–158. [Online]. Available: https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/martinovic

[59] Y. Xiao, Y. Jia, X. Cheng, J. Yu, Z. Liang, and Z. Tian, "I can see your brain: Investigating home-use electroencephalography system security," *IEEE Internet Things J.*, to be published.

[60] G. W. Hart, "Nonintrusive appliance load monitoring," *Proc. IEEE*, vol. 80, no. 12, pp. 1870–1891, Dec. 1992.

[61] L. Stankovic, V. Stankovic, J. Liao, and C. Wilson, "Measuring the energy intensity of domestic activities from smart meter data," *Appl. Energy*, vol. 183, pp. 1565–1580, Dec. 2016.

[62] S. S. Clark, H. Mustafa, B. Ransford, J. Sorber, K. Fu, and W. Xu, "Current events: Identifying webpages by tapping the electrical outlet," in *Proc. Eur. Symp. Res. Comput. Secur.* Springer, 2013, pp. 700–717.

[63] S. S. Clark *et al.*, "WattsUpDoc: Power side channels to nonintrusively discover untargeted malware on embedded medical devices," presented at the USENIX Workshop Health Inf. Technol. Washington, DC, USA: USENIX, 2013. [Online]. Available: https://www.usenix.org/conference/healthtech13/workshop-program/presentation/Clark

[64] M. A. Islam, S. Ren, and A. Wierman, "Exploiting a thermal side channel for power attacks in multi-tenant data centers," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1079–1094.

[65] S. B. Örs, E. Oswald, and B. Preneel, "Power-analysis attacks on an FPGA—First experimental results," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*. Springer, 2003, pp. 35–50.

[66] H. Gamaarachchi and H. Ganegoda, "Power analysis based side channel attack," *CoRR*, 2018. [Online]. Available: http://arxiv.org/abs/1801.00932

[67] Q. A. Chen, Z. Qian, and Z. M. Mao, "Peeking into your app without actually seeing It: UI state inference and novel Android attacks," in *Proc. 23rd USENIX Secur. Symp. (USENIX Security)*. San Diego, CA, USA: USENIX Association, 2014, pp. 1037–1052. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/chen

[68] W. Diao, X. Liu, Z. Li, and K. Zhang, "No pardon for the interruption: New inference attacks on Android through interrupt timing analysis," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 414–432.

[69] X. Zhou *et al.*, "Identity, location, disease and more: Inferring your secrets from Android public resources," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2013, pp. 1017–1028. [Online]. Available: http://doi.acm.org/10.1145/2508859.2516661

[70] Y. Xiao, Y. Jia, X. Cheng, S. Wang, J. Mao, and Z. Liang, "I know your social network accounts: A novel attack architecture for device-identity association," *IEEE Trans. Depend. Sec. Comput.*, to be published.

[71] D. Asonov and R. Agrawal, "Keyboard acoustic emanations," in *Proc. IEEE Symp. Secur. Privacy*, May 2004, pp. 3–11.

[72] L. Zhuang, F. Zhou, and J. D. Tygar, "Keyboard acoustic emanations revisited," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 1, p. 3, Oct. 2009. . [Online]. Available: http://doi.acm.org/10.1145/1609956.1609959

[73] M. Zhou *et al.*, "PatternListener: Cracking Android pattern lock using acoustic signals," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2018, pp. 1775–1787. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243777

[74] L. Cai and H. Chen, "TouchLogger: Inferring keystrokes on touch screen from smartphone motion," in *Proc. 6th USENIX Conf. Hot Topics Security (HotSec)*. Berkeley, CA, USA: USENIX Association, 2011, p. 9. [Online]. Available: http://dl.acm.org/citation.cfm?id=2028040.2028049

[75] Y. Chen, T. Li, R. Zhang, Y. Zhang, and T. Hedgpeth, "EyeTell: Video-assisted touchscreen keystroke inference from eye movements," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 144–160.

[76] L. Sweeney, "K-anonymity: A model for protecting privacy," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, pp. 557–570, 2002.

[77] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," in *Proc. 22nd Int. Conf. Data Eng. (ICDE)*, Apr. 2006, p. 24.

[78] N. Li, T. Li, and S. Venkatasubramanian, "T-closeness: Privacy beyond k-anonymity and l-diversity," in *Proc. 23rd Int. Conf. Data Eng. (ICDE)*, Apr. 2007, pp. 106–115.

[79] K. Rajendran, M. Jayabalan, and M. Ehsan, "A study on k-anonymity, l-diversity, and t-closeness techniques focusing medical data," *Int. J. Comput. Sci. Netw. Secur.*, vol. 17, no. 12, pp. 172–176, Dec. 2017.

[80] C. Dwork, "Differential privacy: A survey of results," in *Proc. Int. Conf. Theory Appl. Models Comput.* Springer, 2008, pp. 1–19.

[81] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Proc. 48th Annu. IEEE Symp. Found. Comput. Sci. (FOCS)*, Washington, DC, USA, Oct. 2007, pp. 94–103.

[82] F. D. McSherry, "Privacy integrated queries: An extensible platform for privacy-preserving data analysis," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, New York, NY, USA, 2009, pp. 19–30. [Online]. Available: http://doi.acm.org/10.1145/1559845.1559850

[83] I. Roy, S. Setty, A. Kilzer, V. Shmatikov, and E. Witchel, "Airavat: Security and privacy for MapReduce," in *Proc. Symp. Netw. Syst. Design Implement. (NSDI)*, Apr. 2010, pp. 1–16. [Online]. Available: https://www.microsoft.com/en-us/research/publication/airavat-security-and-privacy-for-mapreduce/

[84] Q. Xiao, M. K. Reiter, and Y. Zhang, "Mitigating storage side channels using statistical privacy mechanisms," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2015, pp. 1582–1594. [Online]. Available: http://doi.acm.org/10.1145/2810103.2813645

[85] A. Cheu, A. Smith, J. Ullman, D. Zeber, and M. Zhilyaev, "Distributed differential privacy via shuffling," 2018, *arXiv:1808.01394*. [Online]. Available: https://arxiv.org/abs/1808.01394

[86] J. Zhao, J. Zhang, and H. V. Poor, "Dependent differential privacy for correlated data," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2017, pp. 1–7.

[87] D. Kifer and A. Machanavajjhala, "No free lunch in data privacy," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, New York, NY, USA, 2011, pp. 193–204. [Online]. Available: http://doi.acm.org/10.1145/1989323.1989345

[88] D. Molnar, M. Piotrowski, D. Schultz, and D. Wagner, "The program counter security model: Automatic detection and removal of control-flow side channel attacks," in *Proc. Int. Conf. Inf. Secur. Cryptol.* Springer, 2005, pp. 156–168.

[89] T. Zhang, Y. Zhang, and R. B. Lee, "CloudRadar: A real-time side-channel attack detection system in clouds," in *Proc. Int. Symp. Res. Attacks, Intrusions, Defenses*. Springer, 2016, pp. 118–140.

[90] R. Strackx and F. Piessens, "The Heisenberg defense: Proactively defending SGX enclaves against page-table-based side-channel attacks," 2017, *arXiv:1712.08519*. [Online]. Available: https://arxiv.org/abs/1712.08519

[91] Y. Fu, E. Bauman, R. Quinonez, and Z. Lin, "Sgx-Lapd: Thwarting controlled side channel attacks via enclave verifiable page faults," in *Proc. Int. Symp. Res. Attacks, Intrusions, Defenses*. Springer, 2017, pp. 357–380.

[92] J. Tang, A. Korolova, X. Bai, X. Wang, and X. Wang, "Privacy loss in Apple's implementation of differential privacy on MacOS 10.12," 2017, *arXiv:1709.02753*. [Online]. Available: https://arxiv.org/abs/1709.02753

[93] A. Haeberlen, B. C. Pierce, and A. Narayan, "Differential privacy under fire," in *Proc. 20th USENIX Conf. Secur. (SEC)*. Berkeley, CA, USA: USENIX Association, 2011, p. 33. [Online]. Available: http://dl.acm.org/citation.cfm?id=2028067.2028100

[94] S. Lee, M.-W. Shih, P. Gera, T. Kim, H. Kim, and M. Peinado, "Inferring fine-grained control flow inside SGX enclaves with branch shadowing," in *Proc. 26th USENIX Secur. Symp. (USENIX Security)*. Vancouver, BC, Canada: USENIX Association, 2017, pp. 557–574. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/lee-sangho

[95] C. Anley, "Advanced SQL injection in SQL server applications," in *Proc. CGISecurity*, 2002, pp. 1–25.

[96] (2016). *Cisco Fog Director Cross-Site Scripting Vulnerability*. [Online]. Available: https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20160201-fd

[97] M. Martin and M. S. Lam, "Automatic generation of XSS and SQL injection attacks with goal-directed model checking," in *Proc. 17th Conf. Secur. Symp.* Berkeley, CA, USA: USENIX Association, 2008, pp. 31–43.

[98] A. Costin, "IoT/embedded vs. security: Learn from the past, apply to the present, prepare for the future," in *Proc. FRUCT Conf.*, 2017.

[99] M. McIntosh and P. Austel, "XML signature element wrapping attacks and countermeasures," in *Proc. Workshop Secure Web Services*, 2005, pp. 20–27.

[100] A. Greenberg. (2017). *The Reaper IoT Botnet Has Already Infected a Million Networks*. [Online]. Available: https://www.wired.com/story/reaper-iot-botnet-infected-million-networks/(visited on 01/13/2018)

[101] A. Cui, M. Costello, and S. Stolfo, "When firmware modifications attack: A case study of embedded exploitation," in *Proc. NDSS*, 2013, pp. 1–13.

[102] G. Hernandez and D. Buentello, "Smart nest thermostat a smart spy in your home," in *Proc. Black Hat*, 2014.

[103] J. Maskiewicz, B. Ellis, J. Mouradian, and H. Shacham, "Mouse trap: Exploiting firmware updates in USB peripherals," in *Proc. 8th USENIX Workshop Offensive Technol. (WOOT)*. San Diego, CA, USA: USENIX Association, 2014. [Online]. Available: https://www.usenix.org/conference/woot14/workshop-program/presentation/maskiewicz

[104] (2018). *Application Sandbox*. [Online]. Available: https://source.android.com/security/app-sandbox

[105] R. Wang, L. Xing, S. Chen, and S. Chen, "Unauthorized origin crossing on mobile platforms: Threats and mitigation," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, Nov. 2013, pp. 635–646. [Online]. Available: https://www.microsoft.com/en-us/research/publication/unauthorized-origin-crossing-on-mobile-platforms-threats-and-mitigation/

[106] C. Ren, Y. Zhang, H. Xue, T. Wei, and P. Liu, "Towards discovering and understanding task hijacking in android," in *Proc. 24th USENIX Secur. Symp. (USENIX Security)*. Washington, DC, USA: USENIX Association, 2015, pp. 945–959. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/ren-chuangang

[107] Y. Xiao, G. Bai, J. Mao, Z. Liang, and W. Cheng, "Privilege leakage and information stealing through the android task mechanism," in *Proc. IEEE Symp. Privacy-Aware Comput. (PAC)*, Aug. 2017, pp. 152–163.

[108] K. Chen *et al.*, "Following devil's footprints: Cross-platform analysis of potentially harmful libraries on Android and iOS," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 357–376.

[109] T. Li *et al.*, "Unleashing the walking dead: Understanding cross-app remote infections on mobile WebViews," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2017, pp. 829–844. [Online]. Available: http://doi.acm.org/10.1145/3133956.3134021

[110] W. G. J. Halfond, J. Viegas, and A. Orso, "A classification of SQL injection attacks and countermeasures," Tech. Rep., 2006.

[111] Y.-W. Huang, F. Yu, C. Hang, C.-H. Tsai, D.-T. Lee, and S.-Y. Kuo, "Securing Web application code by static analysis and runtime protection," in *Proc. 13th Int. Conf. World Wide Web (WWW)*, New York, NY, USA, 2004, pp. 40–52. [Online]. Available: http://doi.acm.org/10.1145/988672.988679

[112] V. B. Livshits and M. S. Lam, "Finding security vulnerabilities in java applications with static analysis," in *Proc. USENIX Secur. Symp.*, vol. 14, 2005, p. 18. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251398.1251416

[113] P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan, "CANDID: Dynamic candidate evaluations for automatic prevention of SQL injection attacks," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 2, pp. 14:1–14:39, Mar. 2010. [Online]. Available: http://doi.acm.org/10.1145/1698750.1698754

[114] I. Lee, S. Jeong, S.-S. Yeo, and J. Moon, "A novel method for SQL injection attack detection based on removing SQL query attribute values," *Math. Comput. Model.*, vol. 55, nos. 1–2, pp. 58–68, Jan. 2012.

[115] K. A. Jackson and B. T. Bennett, "Locating SQL injection vulnerabilities in Java byte code using natural language techniques," in *Proc. SoutheastCon*, Apr. 2018, pp. 1–5.

[116] K. Ross, M. Moh, T.-S. Moh, and J. Yao, "Multi-source data analysis and evaluation of machine learning techniques for SQL injection detection," in *Proc. ACMSE 2018 Conf.*, New York, NY, USA, 2018, pp. 1:1–1:8. [Online]. Available: http://doi.acm.org/10.1145/3190645.3190670

[117] S. Gupta and B. B. Gupta, "Cross-Site Scripting (XSS) attacks and defense mechanisms: Classification and state-of-the-art," *Int. J. Syst. Assurance Eng. Manage.*, vol. 8, no. 1, pp. 512–530, 2017.

[118] G. Kaur, B. Pande, A. Bhardwaj, G. Bhagat, and S. Gupta, "Defense against HTML5 XSS Attack Vectors: A Nested Context-Aware Sanitization Technique," in *Proc. 8th Int. Conf. Cloud Comput., Data Sci. Eng. (Confluence)*, Jan. 2018, pp. 442–446.

[119] S. Rathore, P. K. Sharma, and J. H. Park, "XSSClassifier: An Efficient XSS Attack Detection Approach Based on Machine Learning Classifier on SNSs," *J. Inf. Process. Syst.*, vol. 13, no. 4, pp. 1014–1028, 2017.

[120] N. Jovanovic, E. Kirda, and C. Kruegel, "Preventing cross site request forgery attacks," in *Proc. Securecomm Workshops*, Aug. 2006, pp. 1–10.

[121] A. Johnson. (May 9, 2006). *The Referer Header*. [Online]. Available: http://cephas.net/blog/2007/02/06/the-referer-header-intranets-and-privacy/

[122] A. Barth, C. Jackson, and J. C. Mitchell, "Robust defenses for cross-site request forgery," in *Proc. 15th ACM Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2008, pp. 75–88. doi: 10.1145/1455770.1455782.

[123] B. S. Y. Fung and P. P. C. Lee, "A privacy-preserving defense mechanism against request forgery attacks," in *Proc. IEEE 10th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Nov. 2011, pp. 45–52.

[124] M. Srokosz, D. Rusinek, and B. Ksiezopolski, "A new WAF-based architecture for protecting Web applications against CSRF attacks in malicious environment," in *Proc. Federated Conf. Comput. Sci. Inf. Syst. (FedCSIS)*, Sep. 2018, pp. 391–395.

[125] M. Jensen, C. Meyer, J. Somorovsky, and J. Schwenk, "On the effectiveness of XML Schema validation for countering XML Signature Wrapping attacks," in *Proc. 1st Int. Workshop Securing Services Cloud (IWSSC)*, Sep. 2011, pp. 7–13.

[126] A. N. Gupta and P. S. Thilagam, "Detection of XML signature wrapping attack using node counting," in *Proc. 3rd Int. Symp. Big Data Cloud Comput. Challenges (ISBCC)*. Springer, 2016, pp. 57–63.

[127] J. Kumar, B. Rajendran, B. S. Bindhumadhava, and N. S. C. Babu, "XML wrapping attack mitigation using positional token," in *Proc. Int. Conf. Public Key Infrastruct. Appl. (PKIA)*, Nov. 2017, pp. 36–42.

[128] B. Lee and J.-H. Lee, "Blockchain-based secure firmware update for embedded devices in an Internet of Things environment," *J. Supercomput.*, vol. 73, no. 3, pp. 1152–1167, 2017. [Online]. Available: https://doi.org/10.1007/s11227-016-1870-0

[129] S. Weiser, M. Werner, F. Brasser, M. Malenko, S. Mangard, and A.-R. Sadeghi, "TIMBER-V: Tag-isolated memory bringing fine-grained enclaves to RISC-V," in *Proc. NDSS*, 2019, pp. 1–15.

[130] *Arm Market*. Accessed: Jan. 14, 2019. [Online]. Available: https://www.arm.com/-/media/global/company/investors/PDFs/Arm_SB_Q1_2018_Roadshow_Slides_Final.pdf

[131] B. Schmerl, J. Gennari, J. Cámara, and D. Garlan, "Raindroid: A system for run-time mitigation of Android intent vulnerabilities [poster]," in *Proc. Symp. Bootcamp Sci. Secur. (HotSos)*, New York, NY, USA, 2016, pp. 115–117. [Online]. Available: http://doi.acm.org/10.1145/2898375.2898389

[132] D.-J. Wu, C.-H. Mao, T.-E. Wei, H.-M. Lee, and K.-P. Wu, "DroidMat: Android malware detection through manifest and API calls tracing," in *Proc. 7th Asia Joint Conf. Inf. Secur. (Asia JCIS)*, Aug. 2012, pp. 62–69.

[133] M. Backes, S. Bugiel, and E. Derr, "Reliable third-party library detection in android and its security applications," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2016, pp. 356–367. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978333

[134] E. Fernandes, A. Rahmati, J. Jung, and A. Prakash, "Decentralized action integrity for trigger-action IoT platforms," in *Proc. Netw. Distrib. Syst. Symp. (NDSS)*, 2018, pp. 18–21.

[135] (2017). *100 mb Password Dictionary*. [Online]. Available: https://github.com/danielmiessler/SecLists/tree/master/Passwords

[136] R. X. Lu and Z. F. Cao, "Simple three-party key exchange protocol," *Comput. Secur.*, vol. 26, no. 1, pp. 94–97, Feb. 2007.

[137] J. Nam, J. Paik, H. K. Kang, U. M. Kim, and D. Won, "An off-line dictionary attack on a simple three-party key exchange protocol," *IEEE Commun. Lett.*, vol. 13, no. 3, pp. 205–207, Mar. 2009.

[138] O. Nakhila, A. Attiah, Y. Jin, and C. Zou, "Parallel active dictionary attack on WPA2-PSK Wi-Fi networks," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2015, pp. 665–670.

[139] T. C. Clancy, N. Kiyavash, and D. J. Lin, "Secure smartcardbased fingerprint authentication," in *Proc. ACM SIGMM Workshop Biometrics Methods Appl.*, 2003, pp. 45–52.

[140] A. Roy, N. Memon, J. Togelius, and A. Ross, "Evolutionary methods for generating synthetic MasterPrint templates: Dictionary attack in fingerprint recognition," in *Proc. Int. Conf. Biometrics (ICB)*, Feb. 2018, pp. 39–46.

[141] A. Cassola, W. K. Robertson, E. Kirda, and G. Noubir, "A practical, targeted, and stealthy attack against WPA enterprise authentication," in *Proc. NDSS*, 2013, pp. 1–15.

[142] K. Bhargavan and G. Leurent, "Transcript collision attacks: Breaking authentication in TLS, IKE, and SSH," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2016, pp. 1–17.

[143] D. Felsch, M. Grothe, J. Schwenk, A. Czubak, and M. Szymanek, "The dangers of key reuse: Practical attacks on IPsec IKE," in *Proc. 27th USENIX Secur. Symp. (USENIX Security)*. Baltimore, MD, USA: USENIX Association, 2018, pp. 567–583.

[144] M. Vanhoef and F. Piessens, "Key reinstallation attacks: Forcing nonce reuse in WPA2," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2017, pp. 1313–1328. [Online]. Available: http://doi.acm.org/10.1145/3133956.3134027

[145] M. Vanhoef and F. Piessens, "Release the Kraken: New KRACKs in the 802.11 standard," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2018, pp. 299–314. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243807

[146] G.-H. Tu, Y. Li, C. Peng, C.-Y. Li, H. Wang, and S. Lu, "Control-plane protocol interactions in cellular networks," in *Proc. ACM Conf. SIGCOMM*,

New York, NY, USA, 2014, pp. 223–234. [Online]. Available: http://doi.acm.org/10.1145/2619239.2626302

[147] D. Rupprecht, K. Jansen, and C. Pöpper, "Putting LTE security functions to the test: A framework to evaluate implementation correctness," in *Proc. 10th USENIX Workshop Offensive Technol. (WOOT)*. Austin, TX, USA: USENIX Association, 2016. [Online]. Available: https://www.usenix.org/conference/woot16/workshop-program/presentation/rupprecht

[148] S. R. Hussain, O. Chowdhury, S. Mehnaz, and E. Bertino, "LTEInspector: A systematic approach for adversarial testing of 4G LTE," in *Proc. NDSS*, 2018, pp. 1–15.

[149] S. R. Hussain, M. Echeverria, O. Chowdhury, N. Li, and E. Bertino, "Privacy attacks to the 4G and 5G cellular paging protocols using side channel information," in *Proc. NDSS*, 2019, pp. 1–15.

[150] E. Hammer-Lahav, *The OAuth 1.0 Protocol*, document RFC 5849, IETF, 2010.

[151] D. Hardt, *The OAuth 2.0 Authorization Framework*, document RFC 6749, IETF, 2012.

[152] E. Y. Chen, Y. Pei, S. Chen, Y. Tian, R. Kotcher, and P. Tague, "OAuth demystified for mobile application developers," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 892–903.

[153] S.-T. Sun and K. Beznosov, "The devil is in the (implementation) details: An empirical analysis of OAuth SSO systems," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 378–390.

[154] Y. Jia, Y. Xiao, J. Yu, X. Cheng, Z. Liang, and Z. Wan, "A novel graph-based mechanism for identifying traffic vulnerabilities in smart home IoT," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 1493–1501.

[155] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in *Proc. 18th ACM Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2011, pp. 627–638. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046779

[156] G. Ho, D. Leung, P. Mishra, A. Hosseini, D. Song, and D. Wagner, "Smart locks: Lessons for securing commodity Internet of Things devices," in *Proc. 11th ACM Asia Conf. Comput. Commun. Secur. (ASIA CCS)*, New York, NY, USA, 2016, pp. 461–472. [Online]. Available: http://doi.acm.org/10.1145/2897845.2897886

[157] B. Pinkas and T. Sander, "Securing passwords against dictionary attacks," in *Proc. 9th ACM Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2002, pp. 161–170. [Online]. Available: http://doi.acm.org/10.1145/586110.586133

[158] A. T. B. Jin, D. N. C. Ling, and A. Goh, "BioHashing: Two factor authentication featuring fingerprint data and tokenised random number," *Pattern Recognit.*, vol. 37, no. 11, pp. 2245–2255, Apr. 2004.

[159] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, Jun. 2015, pp. 815–823.

[160] F. Aloul, S. Zahidi, and W. El-Hajj, "Two factor authentication using mobile phones," in *Proc. IEEE/ACS Int. Conf. Comput. Syst. Appl.*, May 2009, pp. 641–644.

[161] M. Dailey and C. Namprempre, "A text graphics character CAPTCHA for password authentication," in *Proc. IEEE Region 10 Conf. TENCON*, vol. 2, Nov. 2004, pp. 45–48.

[162] N. Karapanos, C. Marforio, C. Soriente, and S. Čapkun, "Sound-proof: Usable two-factor authentication based on ambient sound," in *Proc. 24th USENIX Secur. Symp. (USENIX Security)*. Washington, DC, USA: USENIX Association, 2015, pp. 483–498. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/karapanos

[163] C. Mulliner, R. Borgaonkar, P. Stewin, and J.-P. Seifert, "SMS-based one-time passwords: Attacks and defense," in *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*. Springer, 2013, pp. 150–159.

[164] D. Wang, J. Ming, T. Chen, X. Zhang, and C. Wang, "Cracking IoT device user account via brute-force attack to SMS authentication code," in *Proc. 1st Workshop Radical Experiential Secur.*, 2018, pp. 57–60.

[165] M. Joshi, B. Mazumdar, and S. Dey, "Security vulnerabilities against fingerprint biometric system," 2018, *arXiv:1805.07116*. [Online]. Available: https://arxiv.org/abs/1805.07116

[166] Z. Zhou, D. Tang, X. Wang, W. Han, X. Liu, and K. Zhang, "Invisible mask: Practical attacks on face recognition with infrared," 2018, *arXiv:1803.04683*. [Online]. Available: https://arxiv.org/abs/1803.04683

[167] Y. Wang, Y. Huang, W. Zheng, Z. Zhou, D. Liu, and M. Lu, "Combining convolutional neural network and self-adaptive algorithm to defeat synthetic multi-digit text-based CAPTCHA," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Mar. 2017, pp. 980–985.

[168] B. Shrestha, M. Shirvanian, P. Shrestha, and N. Saxena, "The sounds of the phones: Dangers of zero-effort second factor login based on ambient audio," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2016, pp. 908–919. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978328

[169] D. Boneh, H. Corrigan-Gibbs, and S. Schechter, "Balloon hashing: A memory-hard function providing provable protection against sequential attacks," in *Advances in Cryptology—ASIACRYPT*. Springer, 2016, pp. 220–248.

[170] S. Jarecki, H. Krawczyk, M. Shirvanian, and N. Saxena, "Device-enhanced password protocols with optimal online-offline protection," in *Proc. 11th ACM Asia Conf. Comput. Commun. Security (ASIA CCS)*, New York, NY, USA, 2016, pp. 177–188. [Online]. Available: http://doi.acm.org/10.1145/2897845.2897880

[171] Y. Liu, "Defense of WPA/WPA2-PSK brute forcer," in *Proc. 2nd Int. Conf. Inf. Sci. Control Eng.*, Apr. 2015, pp. 185–188.

[172] J. Noh, J. Kim, G. Kwon, and S. Cho, "Secure key exchange scheme for WPA/WPA2-PSK using public key cryptography," in *Proc. IEEE Int. Conf. Consum. Electron.-Asia (ICCE-Asia)*, Oct. 2016, pp. 1–4.

[173] S. Sivakorn, G. Argyros, K. Pei, A. D. Keromytis, and S. Jana, "HVLearn: Automated black-box analysis of hostname verification in SSL/TLS implementations," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 521–538.

[174] K. Bhargavan, B. Blanchet, and N. Kobeissi, "Verified models and reference implementations for the TLS 1.3 standard candidate," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 483–502.

[175] X. Duan and X. Wang, "Fast authentication in 5G HetNet through SDN enabled weighted secure-context-information transfer," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[176] C. Zhao, L. Huang, Y. Zhao, and X. Du, "Secure machine-type communications toward LTE heterogeneous networks," *IEEE Wireless Commun.*, vol. 24, no. 1, pp. 82–87, Feb. 2017.

[177] J. Ni, X. Lin, and X. S. Shen, "Efficient and secure service-oriented authentication supporting network slicing for 5G-enabled IoT," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 644–657, Mar. 2018.

[178] R. Yang, W. C. Lau, and S. Shi, "Breaking and fixing mobile app authentication with OAuth2.0-based protocols," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.* Springer, 2017, pp. 313–335.

[179] M. Shehab and F. Mohsen, "Securing OAuth implementations in smart phones," in *Proc. 4th ACM Conf. Data Appl. Secur. Privacy (CODASPY)*, New York, NY, USA, 2014, pp. 167–170. [Online]. Available: http://doi.acm.org/10.1145/2557547.2557588

[180] S. Cirani, M. Picone, P. Gonizzi, L. Veltri, and G. Ferrari, "IoT-OAS: An OAuth-based authorization service architecture for secure services in IoT scenarios," *IEEE Sensors J.*, vol. 15, no. 2, pp. 1224–1234, Feb. 2015.

[181] Y. Tian *et al.*, "SmartAuth: User-centered authorization for the Internet of Things," in *Proc. 26th USENIX Secur. Symp. (USENIX Security)*, Vancouver, BC, USA: USENIX Association, 2017, pp. 361–378. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/tian

[182] Z. B. Celik *et al.*, "Sensitive information tracking in commodity IoT," in *Proc. 27th USENIX Secur. Symp. (USENIX Security)*. Baltimore, MD, USA: USENIX Association, 2018, pp. 1687–1704.

[183] I. Bastys, M. Balliu, and A. Sabelfeld, "If this then what?: Controlling flows in IoT apps," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2018, pp. 1102–1119. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243841

[184] Z. B. Celik, G. Tan, and P. McDaniel, "IoTGuard: Dynamic enforcement of security and safety policy in commodity IoT," in *Proc. NDSS*, 2019, pp. 1–15.

[185] Y. J. Jia *et al.*, "ContexIoT: Towards providing contextual integrity to appified IoT platforms," in *Proc. NDSS*, 2017.

[186] R. Schuster, V. Shmatikov, and E. Tromer, "Situational access control in the Internet of Things," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2018, pp. 1056–1073. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243817

[187] J. Wang and Q. Chen, "ASPG: Generating Android semantic permissions," in *Proc. IEEE 17th Int. Conf. Comput. Sci. Eng.*, Dec. 2014, pp. 591–598.

[188] E. Fernandes, J. Paupore, A. Rahmati, D. Simionato, M. Conti, and A. Prakash, "FlowFence: Practical Data Protection for Emerging IoT Application Frameworks," in *Proc. 25th USENIX Secur. Symp. (USENIX Security)*. Austin, TX, USA: USENIX Association, 2016, pp. 531–548. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/fernandes

[189] A. K. Sikder, H. Aksu, and A. S. Uluagac, "6thSense: A context-aware sensor-based attack detector for smart devices," in *Proc. 26th USENIX Secur. Symp. (USENIX Security)*. Vancouver, BC, USA: USENIX Association, 2017, pp. 397–414. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/sikder

[190] Y. Aafer, G. Tao, J. Huang, X. Zhang, and N. Li, "Precise Android API protection mapping derivation and reasoning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 1151–1164.

[191] L. Qian, Z. Luo, Y. Du, and L. Guo, "Cloud computing: An overview," in *Proc. IEEE Int. Conf. Cloud Comput.*. Springer, 2009, pp. 626–631.

[192] (2018). *Top 50 Products by Total Number of 'distinct' Vulnerabilities in 2018*. [Online]. Available: https://www.cvedetails.com/top-50-products.php?year=2018

[193] (2018). *Cloud Computing: Attack Vectors and Counter Measures*. [Online]. Available: https://resources.infosecinstitute.com/cloud-computing-attacks-vectors-and-counter-measures/

[194] M. Hylkema, "A survey of database inference attack prevention methods," Educ. Technol. Res., Tech. Rep., 2009.

[195] D. Barrera, I. Molloy, and H. Huang, "Standardizing IoT network security policy enforcement," in *Proc. Workshop Decentralized IoT Secur. Standards (DISS)*, 2018, p. 6.

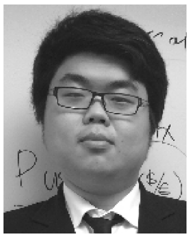
[196] S. Christey and R. A. Martin. (2007). *Vulnerability Type Distributions in CVE*. [Online]. Available: http://cwe.mitre.org/documents/vuln-trends/vuln-trends.pdf

ABOUT THE AUTHORS

Yinhao Xiao received the Ph.D. degree in computer science from The George Washington University, Washington, DC, USA, in 2019, under the supervision of Prof. X. Cheng.

He is currently a Faculty Member with the School of Information Science, Guangdong University of Finance and Economics, Guangzhou, China. He has published work in top international conferences and journals, including the IEEE INFOCOM and the IEEE INTERNET OF THINGS (IoT) JOURNAL. His current research interests include the IoT security, smartphone security, and binary security.

Dr. Xiao has served as a TPC Member at several academic conferences, including the International Conference on Wireless Algorithms, Systems, and Applications (since 2018) and the 2019 International Conference on Blockchain. He has also served as the Web Chair for the 2018 IEEE International Conference on Industrial Internet.



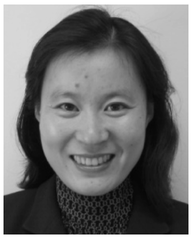
Yizhen Jia received the bachelor's degree in computer science from Beihang University, Beijing, China, in 2015. He is currently working toward the Ph.D. degree with the Computer Science Department, The George Washington University, Washington, DC, USA, under the supervision of Prof. X. (Susan) Cheng.

He has published several technical papers in top international conferences and journals, such as the IEEE INFOCOM and the IEEE INTERNET OF THINGS (IoT) JOURNAL. His current research interests include the Internet of Things, security, and edge computing.

Chunchi Liu received the bachelor's degree (Hons.) in computer science from Beijing Normal University, Beijing, China, in 2017. He is currently working toward the Ph.D. degree with the Computer Science Department, The George Washington University, Washington, DC, USA, under the supervision of Prof. X. (Susan) Cheng.

He has published technical papers in top international conferences and journals, such as the IEEE INFOCOM and the IEEE INTERNET OF THINGS (IoT) JOURNAL. His current research interests include blockchain, Internet of Things, security, and applied cryptography.

Mr. Liu served as a TPC Member for a few conferences, such as the 2019 IEEE Blockchain Conference and the International Conference on Wireless Algorithms, Systems, and Applications 2019, and a reviewer for several distinguished journals, such as the *IEEE Wireless Communications Magazine* and the IEEE TRANSACTIONS ON MOBILE COMPUTING.



Xiuzhen Cheng (Fellow, IEEE) received the M.S. and Ph.D. degrees in computer science from the University of Minnesota–Twin Cities, Minneapolis, MN, USA, in 2000 and 2002, respectively.

She was a Program Director for the U.S. National Science Foundation (NSF) in 2006 (full time) and from 2008 to 2010 (part time). She is currently a Professor with the Department of Computer Science, The George Washington University, Washington, DC, USA. She has published more than 200 peer-reviewed papers. Her current research interests include blockchain computing, intelligent Internet of Things, privacy-aware computing, wireless and mobile security, and algorithm design and analysis.

Dr. Cheng has served/is serving on the Editorial Board of several technical journals, including the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, and IEEE WIRELESS COMMUNICATIONS; and the Technical Program Committee of many professional conferences/workshops, including ACM Mobihoc, ACM Mobisys, IEEE INFOCOM, IEEE ICDCS, IEEE ICC, and IEEE/ACM IWQoS. She has also chaired several international conferences, including ACM Mobihoc 2014 and IEEE PAC 2018. She is the Founder and the Steering Committee Chair of the International Conference on Wireless Algorithms, Systems, and Applications (WASA, launched in 2006).

Jiguo Yu (Senior Member, IEEE) received the Ph.D. degree from the School of Mathematics, Shandong University, Jinan, China, in 2004.

He became a Full Professor at the School of Computer Science, Qufu Normal University, Jining, Shandong, China, in 2007. He is also a Full Professor with the Qilu University of Technology (Shandong Academy of Sciences), Jinan, and the Shandong Computer Science Center (National Supercomputer Center), Jinan. He is interested in designing and analyzing algorithms for many computationally hard problems in networks. His current research interests include privacy-aware computing, wireless networking, distributed algorithms, peer-to-peer computing, and graph theory.

Dr. Yu is a member of the Association for Computing Machinery and a Senior Member of the China Computer Federation (CCF).

Weifeng Lv received the Ph.D. degree in computer science from Beihang University, Beijing, China, in 1998.

He is currently a Professor of computer science, the Dean of the School of Computer Science and Engineering, and the Vice Director of the State Key Laboratory of Software Development Environment, Beihang University. His current research interests include massive information system, urban cognitive computing, swarm intelligence, and smart cities.

Dr. Lv serves as the Vice-President and the Secretary-General of the China Software Industry Association and the Director of the National Engineering Research Center for Science and Technology Resources Sharing Service. He received multiple internationally renowned awards, including the Second Prize of the 2016 China National Science and Technology Invention Award and the First Prize of the 2010 Beijing Science and Technology Award.