XLF: A Cross-layer Framework to Secure the Internet of Things (IoT)

An Wang
Case Western Reserve University
axw474@case.edu

Aziz Mohaisen
University of Central Florida
mohaisen@ucf.edu

Songqing Chen
George Mason University
sqchen@gmu.edu

***Abstract*—The burgeoning Internet of Things (IoT) has offered unprecedented opportunities for innovations and applications that are continuously changing our life. At the same time, the large amount of pervasive IoT applications have posed paramount threats to the user's security and privacy. While a lot of efforts have been dedicated to deal with such threats from the hardware, the software, and the applications, in this paper, we argue and envision that more effective and comprehensive protection for IoT systems can only be achieved via a cross-layer approach. As such, we present our initial design of XLF, a cross-layer framework towards this goal. XLF can secure the IoT systems not only from each individual layer of device, network, and service, but also through the information aggregation and correlation of different layers.**

I. Introduction

The Internet-of-Things (IoT) concept [1], [2] has rapidly emerged as a new computing paradigm, where a great variety of devices are instrumented in a way so that they could be queried and operated over the Internet either directly by the users through an operation panel or by automated programs that encapsulate their behaviors and objectives [3]. This paradigm is continuously shaping the society of tomorrow and improving urban life substantially, highlighted by its contributions in smart cities, health care, and transportation, among others. To a great extend, IoT has revolutionized the way in which individuals and organizations interact with the physical world.

IoT has evolved significantly in the past decade. The concept has expanded and attracted more attention as its potentials to see real-world implementations start to mature. The advancements of networking technologies and the paradigm of Cloud Computing have further made the massive scale of IoT devices a reality. These technologies bring many benefits to the device manufactures. For example, 1) management and operations of the devices could be separated by running management applications, such as data analytic tools, on the remote cloud; 2) due to the simplified operations, the design and implementation of the device hardware become simpler and the devices could be produced at a relatively low cost; 3) Cloud Computing provides a unified interface to enable devices from different vendors to interact easily.

However, many challenges are also emerging along with the explosion of the IoT devices. First, there is a lack of universal standards for the IoT platforms. Second, the existing network infrastructures might not be able to keep up with the growth of the IoT devices. Last but not least, security and privacy remain a big concern for IoT users. Among these challenges, the security and privacy have become imperative with more and more revealed incidents of information leakage and attacks via IoT devices. Several factors make it particularly challenging to address the security issues in IoT environments, including the device heterogeneity, their interoperability, and the layering design of the IoT platforms, etc.

Many efforts have been invested to explore the appropriate approaches for security mechanism design and deployment in the IoT platforms. For example, Farooq *et al.* analyzed the security issues and challenges in IoT architectures by discussing potential violations of data confidentiality, integrity, and availability (CIA) [4]. Ronen *et al.* discovered end exploited a major bug in the implementation of the Touchlink part of the ZigBee Light Link protocol, making it possible for attackers to turn all of a city's lights on or off [5]. The proposed solution was to add hardware support for asymmetric cryptography. Soltan *et al.* demonstrated that an IoT botnet of high wattage devices, such as air conditioners and heaters, could be coordinated to launch large-scale attacks on the power grid [6]. They proposed to prevent such attacks by enhancing the authentication to access the devices. To defend against network attacks launched by IoT devices, Midi *et al.* proposed a knowledge-driven adaptable intrusion detection for the IoT devices, called Kalis [7]. Based on the domain knowledge of different attack behaviors, Kalis detects suspicious network activities from the collected traffic. Miettinen *et al.* built a system to identify the types of different IoT devices and enforce mitigation measures for device-types that have potential security vulnerabilities [8]. The goal is to restrict communications in the network so that adversaries could not connect to the vulnerable devices. Fernandes *et al.* analyzed and revealed security concerns of the current programming framework of smart homes [9]. To address the security vulnerabilities in the framework, Jia *et al.* [10] and Fernandes *et al.* [11] proposed various schemes to monitor the behavior and state transitions of the applications.

However, most of these existing designs either focus on the specific applications (e.g., smart homes) or target at the particular vulnerabilities identified in the hardware or communication protocols. The reality is that with more and more innovative applications and deployment, the attack surface of IoT systems keeps growing, at a very fast pace, and we

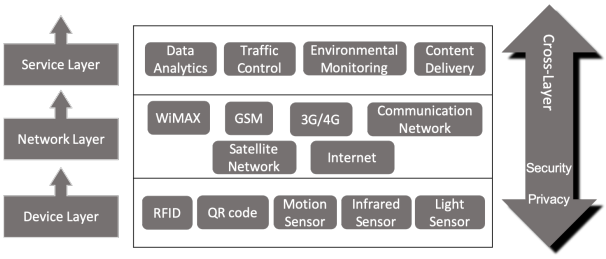

Fig. 1: A generic layered architecture of IoT platforms

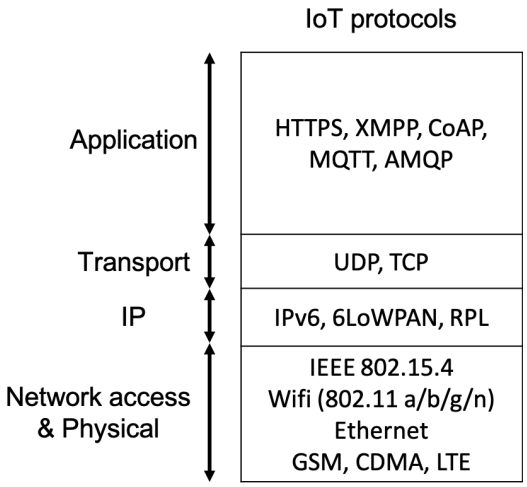

Fig. 2: Some IoT network protocols mapped to the TCP/IP stack

envision that attackers can exploit new vulnerabilities in an ever-increasing attack surface to launch various new attacks.

To this end, we argue that a generic approach would be desired, yet is currently missing. We vision that such a generic approach should take the defense at various layers of the IoT systems into account, and achieve a comprehensive and more effective protection via a cross-layer approach. Thus, in this paper, we present our initial design of XLF, a cross-layer framework to secure the IoT systems, towards this objective. For this purpose, we first present a comprehensive analysis of the IoT system capabilities and attack surface components. Unlike existing approaches that focus on addressing a specific threat, XLF aims to utilize insights into adversarial capabilities as well as the system characteristics at different layers to achieve protection. XLF utilizes various building blocks that can be implemented either at-device, in-network, in the service provider's cloud, or in the gateway.

The remainder of the paper is organized as follows. We provide a layered view of IoT systems in section II. Following this layered approach, we present an analysis of attack surfaces layer by layer in section III. We present our initial design of XLF in section IV. We conclude the paper in section V.

II. Layered View of IoT Systems

IoT systems and applications are very rich and diverse, comprising a variety of computing resources. Table I shows some typical configurations of a few IoT-enabled home appliances, with their computing capabilities.

Regardless of which applications such platforms are designed for, in general, we can see that an IoT system usually consists of three main layers: a device layer, a network layer and a service layer. The device layer provides the front-end interface to the environment and performs some sensing and data collection functions, while the data are usually transmitted (perhaps after aggregation) to the back-end on the cloud or a server where the application runs, through various networking technology. Figure 1 depicts such a layered view of a typical IoT system. As shown in the figure, each layer contains multiple interfaces and different capabilities, which we briefly elaborate in this section.

A. Device Layer

The device layer consists of two often separate sub-layers: a hardware layer (sometimes also called the perception layer) and a resident software layer. The hardware layer provides interfaces for sensors, such as temperature, motion, and smoke sensors, and networking capabilities, such as WiFi, Bluetooth, Z-WAVE, ZigBee, and 6LoWPAN. The resident software layer, on the other hand, runs various Operating Systems (OSes), such as RIOT [12], Contiki [13], and TinyOS [14], and provides a cache to store frequently used OS files or other important files.

B. Network Layer

Communication is central to an IoT system. Networking technologies enable IoT devices to communicate with other devices as well as with applications and services that are running in the cloud or servers. Many emerging and competing networking technologies are being adopted within the IoT space. These technologies come with different constraints, including their communication range, network bandwidth, power usage, interoperability, and security. The Internet relies on standard protocols to ensure secure and reliable communication between heterogeneous devices. Standard protocols specify the rules and formats that devices use for establishing and managing networks, as well as for transmission of data across those networks. Some of the networking protocols are widely adopted within IoT and they could fit in the TCP/IP stack appropriately as we show in Figure 2.

From the security perspective, it is necessary to select networking technologies that implement end-to-end security, such as authentication, encryption, and open port protection. For example, IEEE 802.15.4 includes a security model that provides security features including access control, message integrity, and replay protection. These are implemented by technologies based on this standard such as ZigBee [15]. For authentication, the X.509 standard could be adopted to support authentication, for gateways, users, and applications and services. For wireless network encryption, a Private Pre-Shared Key (PPSK) approach could be employed. Protocols such as Transport-Layer Security (TLS) or Datagram TLS

TABLE I: *Various components in the device layer of a typical home network system; computation, storage, and power limit the security functions that can be implemented on the device.*

Device Type	Chipset	Core Freq.	RAM	Flash Memory	Power
HID Glass Tag Ultra (RFID)	EM 4305	134.2 kHz	512 bit RW	NA	NA
HID Piccolino Tag (RFID)	I-Code SLIx, SLIx-S	13.56Mhz	2048 bit RW	NA	NA
Sensor Devices	Microcontroller	4 - 32Mhz	4 - 16KB	16 - 128KB	Battery
Google Chromecast	ARM Cortex-A7	1.2Ghz	512MB	256MB	NA
NETGEAR Router	Broadcom BCM4709A	1.0Ghz	256MB	128KB	AC Power
Gateway WISE-3310	ARM Cortex-A9	1.0Ghz	NA	4GB	AC Power
REX2 Smart Meter	Teridian 71M6531F SoC	10Mhz	4KB	256KB	Battery
Philips Hue Ligh tbulb	TI CC2530 SoC	32Mhz	8KB	256KB	Battery
Nest Smoke Detector	ARM Cortex-M0	48Mhz	16KB RAM	128KB	Battery
Nest Learning Thermostat	ARM Cortex-A8	800Mhz	512MB RAM	2GB	Battery
Samsung Smart Cam	GM812x SoC	Up to 540Mhz	N/A	Up to 64GB	AC Power
Samsung Smart TV	ARM-based Exonys SoC	1.3Ghz	1GB	N/A	AC Power
OORT Bluetooth Smart Controller	ARM Cortex-M0	50Mhz	16KB/32KB	Up to 256KB	Battery
Dacor Android Oven	PowerVR SGX 540 graphics	1Ghz	512MB	NA	AC Power
Fitbit Smart Wrist Band Flex	ARM Cortex-M3	32Mhz	16KB	128KB	Battery
LG Watch Urbane 2nd Edition	Snapdragon 400 chipset	1.2Ghz	768MB	4GB	Battery
Samsung Watch Gear S2	MSM8x26	1.2Ghz	512MB RAM	4GB	Battery
Apple Watch	S1	520Mhz	512MB RAM	8GB	Battery
iPhone 6s Plus	A9/64-bit/M9 coprocessor	1.85Ghz	2GB	Up to 128GB	Battery
12.9-inch iPad Pro	A9X/64-bit/M9 coprocessor	1.85Ghz	4GB	Up to 256GB	Battery

(DTLS) could ensure privacy and data integrity for communication between applications. Port protection ensures that only ports that are required for communication with the gateway or upstream applications or services remain open to external connections. All other ports should be disabled or protected by applying firewall rules.

C. Service Layer

There are many application domains that are impacted by the emerging IoT devices, including personal and home domain, enterprise domain, utilities domain and mobile domain [3]. The applications are supported by either public or private cloud computing infrastructures. These cloud platforms follow different design paradigms. The extensibility of the framework greatly stimulates device manufacturers and application developers to participate in the ecosystem.

One good example of such platforms is Samsung's SmartThings [16]. The SmartThings architecture provides an abstraction of devices from their distinct capabilities and attributes in a way that allows developers to build applications. A SmartThings Hub is utilized to mediate the communication between the connected devices, the SmartThings cloud and the SmartThings mobile application. On the cloud side, SmartThings cloud consists of several subsystems, including the Connectivity Mangement system, Device Handlers, the Subscription Mnagement system and the SmartApp execution environment. SmartApps executes in a sandboxed environment in the SmartThings cloud and encompasses the interoperabilities of different devices.

Another paradigm that further expands the idea of interoperability is exemplified by a free web-based service, called If This Then That (also known as IFTTT). IFTTT allows users to write trigger-action programs that connect numerous services, social media sites, and physical devices [17]. It has gained a growing popularity since it was launched in 2011. Services are the basic building blocks of IFTTT. They mainly describe a series of data items from a certain web service or actions controlled with certain APIs.

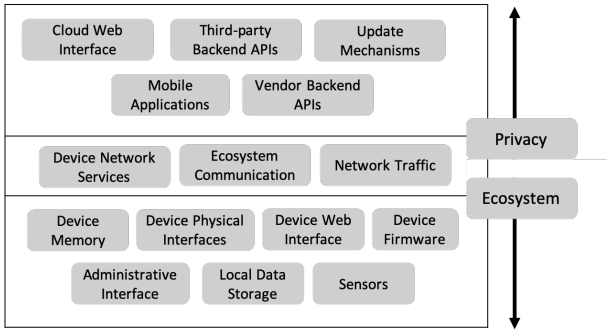

Fig. 3: IoT Attack Surface Areas

III. Attack Surfaces of IoT Systems

The realm of IoT keeps expanding daily. As we discussed before, in such a multi-service and heterogeneous environment, IoT has become a big contributor to security and privacy with its rapidly growing attack surfaces. The Open Web Application Security Project (OWASP) [18], an online community that produces freely-available resources in the field of web application security, suggests a list of attacks that should be understood by manufactures, developers, security researchers, and those looking to deploy or implement IoT technologies within their organizations.

From a system perspective, we classify these attacks into the layered architecture as we presented in the previous section. Figure 3 illustrates which layer these attacks are mapped to. As we can see, except for the privacy and ecosystem concerns, the threats come from every layer of the IoT architecture. In the following, we provide a preliminary attack surface analysis, by following the three layers in our system model.

A. Attacks in the Device Layer

Ideally, devices in the device layer should provide several guarantees: i) have a secure software, ii) allow for authorized access only, and iii) securely store and transmit data to other

TABLE II: *Attack surface: enumeration of vulnerabilities, attacks, and impact at the device layer.*

Device	Vulnerability	Attack	Impact
Smart light bulb	Static password	MitM, password stealing	Bulb controlled by remote
Wall pad	Buffer overflow	Value manipulation, shellcode exe.	Housebreaking, monitoring
Network camera	Firmware integrity	Firmware modulation	damage peripherals
Chromecast	Rickrolling	D/C & reconnects to attacker	Privacy violation.
Coffee machine	Unprotected channel	Listens to UPNP.	Hijack password of Wi-Fi
Fridge	Generic auth.	Malicious code infection	Send malicious mail
Oven	unsecured Wi-Fi	MitM attack	Access other devices

devices and the service clouds. As a result, the attack surface in the device layer is a result of the violation of one or more of those requirements.

For example, software could be vulnerable due to poor software design and language use, including buffer overflow, injection vulnerabilities, failing to handle error correctly, allowing for cross-site scripting, improper use of security packages (such as TLS and SSL at the device layer), and poor usability, among others. They could also come from firmware vulnerabilities, such as firmware downgrade and outdated firmware. Users could access the IoT devices through multiple interfaces, including administration and web interfaces. The violations of administration authentications could result from credential management vulnerabilities, including username enumeration, weak passwords, known default credentials and insecure password recovery mechanisms, as well as insecure direct object references. Attackers could also access the devices from web interfaces. In a recent study performed by Costin *et al.* , they found serious vulnerabilities in at least 24% of the web interfaces of IoT devices, including 225 high impact vulnerabilities by automatic analysis [19]. The devices being tested include routers, DSL/cable modem, VoIP phones and IP/CCTV cameras. In their study, it is revealed that these web interfaces could be leveraged by SQL injection, cross-site scripting, cross site request forgery, command injection and HTTP response splitting. Finally, information leakage is very likely to happen if the devices store unencrypted data or data encrypted with discovered keys within its local storage. This becomes even worse since most devices lack proper data integrity checks and encryption/decryption key management system. Table II shows some examples of the vulnerability, attack method, and impact in the devices.

Manufactures are aware of the potential problems in the devices. For example, some of them bond together to create the Internet of Things Security Foundation to promote security practices [20]. Members include important players such as Siemens, Vodaphone, Webroot, British Telecom (BT), etc. While this sheds a light on some common efforts, however, it will be challenging to set standards for the industry.

B. Attacks in the Network Layer

The networking capability is a key for attacks. The vulnerabilities in the network layer relate to the network services that could be exploited to access the IoT device that might allow an attacker to gain unauthorized access to the device or associated data. More importantly, the devices could be recruited by the attacker to form an army to launch massive scale attacks against some targets. Specific security vulnerabilities include poor implementations of encryption algorithms, open ports via Universal Plug and Play (UPnP), vulnerable UDP services, and lack of payload verification and integrity checks. Besides the issues with the devices themselves, the transportation channel for data could also cause problems. For example, if the data is exchanged with the IoT devices in an unencrypted format, this could easily lead to an attacker sniffing the data and either capturing this data for later use or compromising the device itself. Misconfigurations or bad implementations of SSL/TLS could lead to such vulnerability as well.

C. Attacks in the Service Layer

Since IoT devices often lack the ability to perform complex computations, they typically rely on a back-end cloud or server to provide certain services, such as collaborative tasks among different devices. By doing so, the IoT devices have inherent and implicit trust of the cloud or mobile applications running in a remote platform. If such a platform lacks strong authentication and access control mechanisms, it is very likely that the platform will suffer from injection attacks. As a result, the cloud could be compromised to execute hidden services, such as collecting personally identifiable information (PII) and running misbehaving applications that could lead to unexpected security compromises.

Another important function of backend cloud is to distribute updates to its connected hardware devices. This is usually achieved through a mechanism called over-the-air (OTA) update. This mechanism allows remote update of the Internet-connected hardware devices with new settings, software, and/or firmware. A robust OTA update mechanism is a core part of a system's architecture and a key step to guarantee the security of the devices. It is the device's responsibility to identify and apply the updates to itself. However, if the update is sent unencrypted or unsigned, or the implementations of the verification are not robust, then the device could be easily compromised and controlled by an attacker.

IV. XLF: A Cross-Layer Security Framework

Different from existing defenses that are often designed by targeting specific application or motivated by a particular vulnerability, in this section, we present our vision of XLF, a generic security framework to secure the IoT systems.

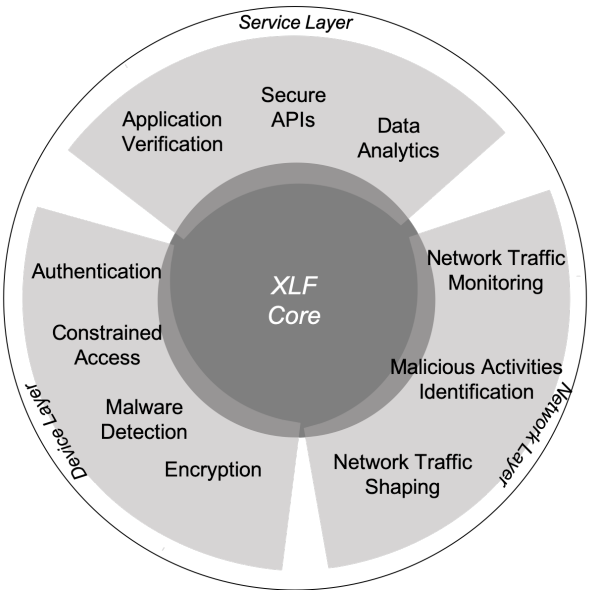

Fig. 4: XLF Cross Layer Security Design

XLF achieves effective protection not only by relying on the various functions that we propose to build in each layer, but also by enhanced protection capability via a cross-layer approach. These functions built in each layer are motivated by our understanding of the IoT system architecture, the system capabilities, and the corresponding attack surface that we have discussed in each layer.

Figure 4 sketches the key components of XLF and the functions we propose to build in each layer. As shown in the figure, we propose to build various functions and capabilities in the device layer, the network layer, and the service layer, respectively, and the details of which shall be discussed soon. Different from previous approaches, while building such capabilities, these layers do not work individually, but interact with each other whenever possible through the XLF Core in the center. The XLF Core not only facilitates the interactions between different layers, but also aggregates the raw and the detection results whenever necessary from each layer, and conducts its own comprehensive evaluations by utilizing the most advanced techniques, such as deep learning. In this way, the cross-layer design of XLF aims to provide proactive protection against intrusions and comprehensive anomaly detection through the combined security functions deployed in each component of the system. In the following, we discuss the proposed functions in each of the three layers and the XLF Core. Note that XLF Core can be implemented independently or integrated with one of the three layers.

While the proposed security mechanisms for each layer are highlighted in Figure 4, in the subsequent sections, we discuss some design details. Although our discussion follows the layered structure, various functions and factors from more than one layer are considered in our discussions.

A. Security Mechanisms in Device Layer

Based on the analysis of attack surfaces, we argue that the following defenses are necessary: authentication, constrained access, malware detection, and encryption. At the device layer, these security functions are designed to secure the firmware and the software running in the IoT devices and their interactions with the end users, the back-end cloud, and the third-party services. In the IoT environment, security threats could also come from physical interactions, such as micro probing and reverse engineering that cause security problems by directly tampering with the hardware components. Since these threats require close proximity to the devices and special tools, they are relatively hard to perform. Therefore, currently we do not consider the physical threats in XLF. In the following, we elaborate on the four major proposed security functions.

1) Authentication: A well-designed authentication method ensures authenticity so that only the authorized personnel or object has access to the private information. While many of the traditional algorithms and authentication methods could still address some security issues, they run into trouble when applied to the IoT environment. One of the problems to be addressed is to make it more convenient for users to perform authentication for multiple accounts following the repeated steps. Two-factor authentication (2FA) has been widely adopted in many services. Some services also use single sign-on (SSO), an authentication system that allow users to sign on to a single account, such as Google or Facebook, and use the same authentication token to access other services. NIST has recently released a new draft of the Security and Privacy Controls, which acknowledges the benefits of combining multi-factor authentication (MFA) and SSO to improve system security [21].

Barreto *et al.* proposed an authentication model that works in two different modes depending on the privilege of the users [22]. For basic users who only access the devices, their requests are authenticated and delegated by the IoT cloud provider. Thus, they get the processed data sent back from the cloud provider. For advanced users who can perform firmware update, their initial authentication is performed by the cloud provider. Then, the user is redirected to the IoT device by means of an SSO authentication so that all the subsequent access requests are directly handled by the device. The proposed schemes fall short in two aspects. First, this solution does not scale to deal with a large number of users with multiple devices. It also increases the latency for users to access their devices. Secondly, due to the resource constraints, it could be challenging for IoT devices to perform the SSO authentication and the proper timestamp validations for the advanced users.

We propose to address these challenges by leveraging the capabilities of the XLF Core to delegate the authentication function for the IoT device. The delegation could be adopted in both the device layer and the network layer, such as a smart gateway. But we envision the delegation proxy having mulitple

TABLE III: *Lightweight cryptographic algorithms.*

Algorithm	Key Size	Block Size	Structure	No. of Rounds
AES	128/192/256	128	SPN*	10/12/14
HEIGHT	128	64	GFS+	32
PRESENT	80/128	64	SPN	31
RC5	02040	32/64/128	Feistel-	1255
TEA	128	64	Feistel	64
XTEA	128	64	Feistel	64
LEA	128,192,256	128	Feistel	24/28/32
DES	54	64	Feistel	16
Seed	128	128	Feistel	16
Twine	80/128	64	Feistel	32
DESL	54	64	Feistel	16
3DES	56/112/168	64	Feistel	48
Hummingbird	256	16	SPN	4
Hummingbird2	256	16	SPN	4
Iceberg	128	64	SPN	16
Pride	128	64	SPN	20

* SubstitutionPermutation Network + Generalized Feistel Structures
- Feistel is a design model for many different block ciphers.

access channels, such as ZigBee, Z-WAVE, and LAN, and being equipped with more computation power and memory resources than the IoT devices. For this purpose, the proxy needs to perform, i) caching of SSO tokens from the cloud provider, ii) SSO authentication and timestamps validation and iii) processing of raw data for low-privileged users. Furthermore, we propose to distinguish access requests from LAN and WAN to enforce different levels of authentication. To support this requirement, we enable the XLF Core to correlate and integrate the authentication results from the delegation proxy and the service cloud. In this case, the proxy authenticates the LAN requests while the cloud service authenticates the WAN request combining both SSO and MFA mechanisms. The XLF Core determines the lifetime of the authentication tokens based on the correlation results. The interactions with the XLF Core capabilities not only address the scalability issue, but also provide a chance for the heterogeneous IoT devices to share a uniform security framework.

2) Encryption: To protect user privacy and the critical information (e.g., username and password), the data exchanged between the IoT devices and the cloud provider should be properly encrypted. Conventional encryption schemes provide strong security guarantees. However, it is challenging to apply them to the resource-limited devices. To address this issue, NIST outlined a number of lightweight cryptography methods that could be useful in IoT devices [23]. In this report, they specified several performance metrics for both hardware and software. They summarized several lightweight cryptography primitives that have been proposed, including lightweight block ciphers, lightweight hash functions, lightweight Message Authentication Codes (MAC), and lightweight stream ciphers. Some of the lightweight cryptographic algorithms are listed in Table III The proposed lightweight algorithms need to be adopted by the vendors to provide end-to-end data security and integrity. However, applying encryption alone can hardly preserve user privacy. We propose a remedy scheme leveraging the information obtained from the network layer to address this issue that will be discussed soon.

3) Constrained Access: This protection is to constrain the resources and third-party services the devices are supposed to communicate with, or access without interrupting their normal operations. One of the front-line defenses should be network access control (NAC). With NAC, the network access requests are either accepted or denied based on a pre-determined set of parameters and policies that are programmed into the system. While the concept is straightforward, it is more challenging to deploy NAC because of its requirements of interacting with the protocols. Such a mechanism could be enforced in either devices or in specifically configured network devices. More often, the devices are hard-coded to connect to certain corporate domains. However, this makes them vulnerable to DNS cache poisoning attacks. In a recent work, Apthorpe *et al.* revealed that passive network observers could infer device types and user behaviors by collecting DNS queries [24]. Thus, DNS plays a vital role in IoT security efforts.

Domain Name System Security Extensions (DNSSEC) is a suite of IETF specifications for securing the information provided by the DNS. While DNSSEC guarantees secure naming, it only has cryptographically signed but not encrypted responses. In addition, DNSSEC has not been widely adopted by mainstream IoT device vendors [25]. Even with the encryption mechanisms adopted by the devices, they are often utilized over TCP connections instead of UDP communications.

There have been many efforts aiming to raise awareness of this issue and provide resources to mitigate these threats. For example, the DNS Privacy Project [26] lists some of the current solutions, such as DNS-over-TLS (DoT), DNS-over-HTTP (DoH), DNS-over-DTLS, DNSCrypt, DNS-over-HTTPS (proxied), etc. But all these protocols are designed for conventional devices with abundant resources. On the other hand, we could not simply adopt the lightweight cryptographic algorithms proposed for the IoT encryptions because that also requires a forklift upgrade of all the DNS servers. Thus, we propose to take advantage of the computing powers supported by XLF Core to bridge the gap between the lightweight cryptography based DNS and the existing DNS privacy solutions. For this purpose, the XLF Core allows the adoption of lightweight encryption in the device layer and the standard encryption mechanisms in the network layer, providing stronger privacy guarantees across the Internet.

4) Malware Detection: Besides the interactions with the external services, the internal health of IoT devices is also very crucial. Once a device gets compromised, all the proposed security mechanisms would fail. In the worst case, other IoT devices connected on the LAN could also be affected. To mitigate such threats, both proactive and reactive mechanisms should be employed. For the proactive mechanisms, all the firmware and software updates should be examined by performing either deep packet inspection or fingerprint identifications. For the reactive mechanisms, malicious activities should trigger alert and be blocked when they happen. Both methods require cooperation and support from the network layer. Such cooperations are facilitated by the XLF Core functions, which we will discuss next.

B. Security Mechanisms in Network Layer

The network layer provides channels for the IoT devices to interact with other entities on the Internet. At the same time, it also provides a pathway for malicious actors. Thus, the security functions in this layer should satisfy several requirements: i) robust to zero-day vulnerabilities, ii) scale to a large number of connected devices, and iii) easy to manage and customize. Based on these considerations, we propose to provide a Security-as-a-Service platform in the XLF Core so that security functions are easy to integrate and interoperate. Such a platform is not necessarily vendor-specific, making it easier to implement custom security functions. One such example is the Samsung SmartThings Shield for Arduino, which provides a way to allow access to the SmartThings platform for home automation projects [27]. Although this was not designed for security purposes, the idea could be extended to accommodate security functions. In this section, we highlight the design of several critical security functions in this layer.

1) Network Traffic Shaping: In the investigation conducted by Apthorpe *et al.* , a network observer could easily infer sensitive device information following three steps [24]. First, network traffic could be separated into several packet streams by the external IP addresses of the cloud or third-party services. Thus, observers could identify and count the distinct clients (IoT devices) behind a Network Address Translator (NAT). In the second step, observers could identify each individual IoT device by associating DNS queries with each packet stream. For example, the Nest Cam queried domains from dropcam.com, which is the predecessor of the Nest Cam. Finally, simple calculations of send/receive rates of each stream reveal potential user interactions with the devices.

Zhang *et al.* took a step further to infer the state of the devices by collecting and fingerprinting the wireless packets exchanged between the IoT devices and the hub device [28]. The fingerprint of an event is defined by a cluster of packet sequences that are similar with each other. Each wireless packet in the sequence is represented by a quadruple of packet length, timestamp of the packet, source device, and destination device. Then, the similarities of the sequences are measured with Levenshtein Distance [29]. Both studies revealed that the network traffic patterns are directly associated with the state of the devices. To defend against such threats, we propose to embed a lightweight yet effective network function in this layer, called traffic shaping.

Traffic shaping is designed to be a bandwidth management technique to optimize or guarantee performance, improve latency, or increase usable bandwidth. In our proposed architecture, it is used to rearrange the packet sequence to preserve user privacy from both external and internal observers. For this purpose, there are two basic operations to execute in our design. First, it should change the packet transmission rates of different flows by inserting random delays. Secondly, for the incoming traffic, redundant packets could be inserted without changing the states of the devices. The existing algorithm could balance the adversary confidence and the bandwidth overhead when performing traffic shaping [30]. But it is not sufficient to address the potential privacy violations in the device. To achieve this goal, we could leverage the data collection and analyzing capability of the XLF Core to employ the adversarial machine learning techniques to perform the traffic shaping.

2) Network Traffic Monitoring: Certain security functions in the device layer rely on the information obtained from the network layer to take actions, such as malware detection. This function is mainly proposed to prevent IoT devices from being infected by malware from unknown or compromised third-party service providers. In a recent work, Alhanahnah *et al.* proposed a method to generate signatures for IoT malwares using extractable strings and N-gram text analysis [31]. In this analysis, they pointed out that there are specific shell commands and IP address strings that are used for Communication and Control (C&C) channel in the malware samples. These keywords could be organized together to generate malware detection rules. Each rule then is used to describe an attack and it contains one or more keywords to be matched in the traffic, offset information for each keyword, and sometimes regular expressions.

Since malicious traffic enters the system through network, the security checks are naturally to be executed in the network layer. However, given the rule set, it is still challenging to perform the keywords matching due to the proposed encryption mechanisms in the device layer. Conventionally, some systems propose to mount a man-in-the-middle attack on SSL by installing fake certificates on the middlebox [32]. However, this breaks the end-to-end security of SSL. Alternatively, we propose to adopt searchable encryption mechanisms to match the keywords in the payload against the rules, similar to BlindBox [33]. To preserve end-to-end security of SSL, a separate secure connection needs to be established between the XLF Core and the service layer so that the XLF Core could perform searching over the encrypted contents. The challenge is the generation of rule sets from the feature keywords we uncovered. This solution also brings significantly extra traffic load to the public Internet considering the number of connected IoT devices. But this could be alleviated by the proposed authentication function that grants different permissions for users with different privileges. As a result, only service providers with advanced SSO tokens are allowed to perform software or firmware updates. Thus, only their traffic is examined through this function.

3) Malicious Activity Identification: This function provides another layer of protection in case the devices are infected. Nokia recently released an analysis of the IoT botnet activity and its evolution since Mirai [34]. In this report, IoT botnets accounted for 78% of the malware carrier network activity detected in 2018. The malicious activities involve hacking activities, scanning activities,and Distributed Denial of Service (DDoS) attacks. Such attacking activities have been extensively studied and many research efforts have aimed to mitigate these attacks over the past decade. The number of

the connected devices makes it almost impossible to provision any effective defense mechanism close to the target. On the other hand, a security mechanism on the device side may lack sufficient information to identify specific attack activities.

Besides the proposed Constrained Access function, we could also rely on the fact that IoT devices present certain behavior patterns for the normal operations. As pointed out in HoMonit proposed by Zhang *et al.* , the states of the devices could be profiled by packet sequences [28]. Furthermore, the state transitions are dictated by the automation programs installed in the service cloud. Therefore, a Deterministic Finite Automation (DFA) could be used to reflect normal device behaviors. Even for devices without automation programs, such as Amazon Echo, their activity patterns should still be predictable since they are not as multi-functional as mobile devices or general purpose machines. Thus, we propose to employ the advanced machine learning techniques in the XLF Core to aggregate the information collected from the service layer and the device layer to learn the normal operations of the IoT devices and trigger alarms when there are considerable deviations.

C. Security Mechanisms in Service Layer

As we discussed before, modern IoT applications are often supported by back-end services running on clouds. They provide core services like remote administration, alerts, and content. Most IoT devices have trust of the service providers by default. However, the cloud endpoints are discovered to have several potential vulnerabilities, including misbehaving services, weak authentications, and insecure APIs, from previous studies. For example, Obermaier *et al.* analyzed four video surveillance systems and found that attackers could inject footage, trigger false alarm, and carry out denial-of-service attacks against the camera system [35]. These threats come from weak authentications, misconfigurations of the infrastructure, and insecure APIs. The authentication could be improved by employing SSO authentications proposed in the device layer and a robust identity management system. We elaborate on the proposed security functions to address the other problems next.

1) Secure APIs: APIs are the focal point of cloud innovation and enable the connections and data sharing. All the applications providing services to IoT devices rely on APIs to function or grow. For example, the cloud IoT applications use APIs to gather data, or even control other devices. However, they remain the most overlooked threat in the IoT environment. This is due to the fact that API vulnerabilities are not easy to spot and require specialized technology for detection and prevention. The most common APIs in the IoT environment are Web APIs and cloud backend APIs.

Simple Object Access Protocol (SOAP) and Representational State Transfer (REST) are two popular approaches for implementing APIs. The built-in standards and envelope-style of payload transport of SOAP require more overhead compared with other API implementations, such as REST. Unlike SOAP, which requires parsing and routing for each request, REST leverages standard HTTP requests. For example, Samsung SmartThings Cloud utilize REST APIs to control and get status notifications from IoT devices. However, SOAP could provide more comprehensive security and compliance benefits.

To secure the APIs against attacks, proper authentications and authorizations are required. For authentications, OAuth2 [36] and OpenID Connect [37] are the most widely adopted mechanisms. Similar to our proposed scheme in the device layer, users should be prevented from accessing API functions outside their predefined roles so that a read-only API client should not be allowed to access an endpoint providing administration functionality. Thus, each API call should be assigned an API token to validate incoming queries and prevent attacks on endpoints.

2) Application Verification: IoT applications are automation programs that gather data from IoT devices and use the information to control and interoperate IoT devices. Several companies provide cloud-backed and programming platforms for users to setup and develop IoT applications, such as Samsung's SmartThings Cloud, Apple's HomeKit, and Google's Weave/Brillo. However, these applications could be vulnerable to attacks due to the design flaws.

Fernandes *et al.* conducted a security analysis of the programming framework of smart homes, particularly focusing on evaluating Samsung's SmartThings platform [9]. Through their analysis, they discovered security-critical design flaws in the capability model and the event subsystem of the SmartThings cloud. The potential security violations caused by the flaws include: over-privileged access in SmartApps, insufficient sensitive event data protection, insecurity of third-party integration, unsafe use of Groovy dynamic method invocation, and unrestricted API access control. Some of the vulnerabilities could be addressed with the above mentioned mechanisms to secure APIs, while other concerns like over-privileged access and insufficient event data protection require more efforts to detect and mitigate.

The over-privileged accesses enable the SmartApps to gain the access to all capabilities of the devices so that a malicious application could run hidden services or even take control of the whole device. The lack of protection of sensitive event data makes it possible to leak sensitive user information to other applications subscribing to the same event. Furthermore, since the integrity of the events is not protected, malicious actors could easily launch spoofing event attacks. Many research efforts have been invested to mitigate such threats. For example, Fernandes *et al.* proposed a framework to support flow policy rules for IoT applications [11]. Jia *et al.* proposed a system to gather information before a sensitive action is executed, and ask for user approval through frequent run-time prompts [10]. Some researchers also explore to address the root cause of these threats. He *et al.* studied the limitations of the current access control and authentication model, and they envisioned a capability-based security model for these platforms [38].

However, most existing solutions are proposed to be executed and enforced in the back-end cloud platform, which will

become unreliable once the cloud gets compromised. Thus, we propose to build a more robust monitoring system on the user end to perform integrity checks against the applications. Since the state transitions of the devices are dictated by the commands received from the applications, monitoring and profiling the state transition patterns could be applied to achieve this goal. To this end, we propose to enable the network layer to collect device status. Then, by employing machine learning techniques, such as time series modeling, the XLF Core could verify that the applications are executing correctly. Furthermore, the smart gateway could provide programming APIs for users to develop their own security applications to deal with the emerging security threats.

3) Data Analytics: Many big data analytic frameworks and models have been proposed to improve the performance and quality of the provided cloud service. Naturally, the data generated by the devices are valuable sources for security analytics. The behaviors of IoT devices are usually dictated by the automation programs, thus the state transitions should follow certain behavior model. Even though the devices are subject to the influence of the physical environment, the devices are often deployed in a static environment. Thus the collected data should present predictive patterns as well. Base on these observations, multi-dimensional security analytics that correlate data from multiple domains help service providers identify anomalies that might be suspicious, malicious, or inadvertent, and provide context intelligence regarding the nature of the threat, threat vectors used, associated business risk, and recommended mitigation.

For example, an automation application could connect smart thermometer with the smart lock on the window by enforcing a policy which is to open the window when the temperature increases above 80°F. Then, if an attacker is in the proximity of the device, it is very easy for the attacker to exploit this policy and increase the temperature of the environment. In such a situation, the cloud service should be able to monitor the state transitions of the device and associate the transitions with the behavior and pattern learnt from history as well as third-party information, such as weather report and the mobility or location of user cellphones. Another utilization of security analytics is to detect whether there has been a spike in CPU on the sensor or irregular amounts of keep-alive packets on the device. These data could indicate whether the device has exceeded its baseline of data or is performing its programmed behaviors.

When combined with threat intelligence data, security analytics help more effectively detect threats. Thus, we propose to build a data analytics module for security purposes in the cloud framework to help deploy strategic mitigation solutions.

D. XLF Core

XLF Core connects and correlates the security functions in different layers by providing delegation function, correlation of the authentication results from different layers, and computing power to bridge the gap between different layers. In addition to the interactions with each layer, there are several capabilities that are central to the design of XLF Core.

IoT ecosystem consists of numerous sensors and services that continuously collect enormous amounts of data. The velocity at which data is collected and processed is higher than it is for the conventional big data analytics. The data collected from different layers are highly structured but also heterogeneous. The integrated analysis of multiple data sources allows to gain important insights in a wide range of security applications, such as malware detection and anomaly detection. However, the integration of various sources of information remains a big challenge due to the need to develop generic methods to take the data heterogeneity into account. For this purpose, we propose to integrate a multi-kernel learning (MKL) module into XLF Core to correlate data from different sources and perform classifications to identify malicious activities. The benefits of applying MKL include: i) it provides a technically sound way to combine features from heterogeneous sources, ii) the feature combination and classifier training could be done simultaneously, making it very efficient, iii) the classifier generates good learning bounds.

Another important learning module we envision to deploy in XLF Core is graph-based learning, which provides a principled approach to community detection. Such techniques have been extensively used in mining social networks and hierarchical image clustering etc. In the IoT ecosystem, users running the same IoT devices and similar automation applications could be considered as a group or community, which should present similar behaviors. Thus, XLF Core should leverage the knowledge obtained from the group to perform data correlations. Particular signals could be associated with certain events through such correlations. The employment of the graph-based learning is particularly useful to provide comprehensive protections for the small IoT environments.

The logic design of XLF Core does not limit the location where it is deployed. But based on practical considerations, it could realize its full potential when deployed in the network layer by extending the existing smart IoT gateway or deployed in the service layer leveraging the computing power of cloud. The proliferation of IoT and with the vast amount of data it produces have pushed the emerging of a new computing paradigm, Edge Computing [39]. Following this paradigm, many previous work have proposed to deploy data analytics platforms at the edge [40], [41]. The deployment of XLF Core at the edge devices, e.g., smart IoT gateway, is also in align with this trend and will benefit from the advances in this area.

V. Conclusion and Ongoing Work

The IoT paradigm provides us unprecedented opportunities for innovations and also challenges. Facing the increasing threats of security and privacy, the community has come together with various defense mechanisms. Instead of customizing defenses for particular applications or addressing particular vulnerabilities, in this paper, we envision that a generic security framework would be more effective by taking a cross layer approach. Accordingly, we have presented our

initial design of XLF to secure IoT systems. We hope that our design can motivate more enhancement and better protection of current and future IoT systems. In the meanwhile, we are actively working on different modules in each layer as we have discussed and implementing the XLF Core. We expect to obtain and publicize some practical results via the experiments with our prototype.

VI. Acknowledgement

We appreciate the constructive comments from reviewers. This work is partially supported by NSF grant CNS-1524462. Aziz Mohaisen was supported in part by NRF grant 2016K1A1A2912757 and a Cyber Florida Seed Grant. An Wang was supported in part by the Institute for Smart, Secure and Connected Systems at Case Western Reserve University through a grant provided by the Cleveland Foundation.

References

[1] K. Ashton *et al.*, "That internet of things thing," *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009.

[2] "Home - oma specworks," https://www.omaspecworks.org/.

[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[4] M. U. Farooq, M. Waseem, A. Khairi, and S. Mazhar, "A critical analysis on the security concerns of internet of things (iot)," *International Journal of Computer Applications*, vol. 111, no. 7, 2015.

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

[6] S. Soltan, P. Mittal, and H. V. Poor, "Blackiot: Iot botnet of high wattage devices can disrupt the power grid," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 15–32.

[7] D. Midi, A. Rullo, A. Mudgerikar, and E. Bertino, "Kalisa system for knowledge-driven adaptable intrusion detection for the internet of things," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 656–666.

[8] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "Iot sentinel: Automated device-type identification for security enforcement in iot," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 2177–2184.

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

[10] Y. J. Jia, Q. A. Chen, S. Wang, A. Rahmati, E. Fernandes, Z. M. Mao, A. Prakash, and S. J. Unviersity, "Contexlot: Towards providing contextual integrity to appified iot platforms." in *NDSS*, 2017.

[11] E. Fernandes, J. Paupore, A. Rahmati, D. Simionato, M. Conti, and A. Prakash, "Flowfence: Practical data protection for emerging iot application frameworks." in *USENIX Security Symposium*, 2016, pp. 531–548.

[12] —, "Riot - the friendly operating system for the internet of things," https://riot-os.org/, April 2019.

[13] ——, "Contiki: The open source operating system for the internet of things," http://www.contiki-os.org/.

[14] ——, "Github - tinyos/tinyos-main: Main development repository for tinyos (an os for embedded, wireless devices)." https://github.com/tinyos/tinyos-main.

[15] ——, "IEEE 802.15.4 - Wikipedia," https://en.wikipedia.org/wiki/IEEE_802.15.4.

[16] ——, "Smartthings. add a little smartness to your things." https://www.smartthings.com/.

[17] B. Ur, M. Pak Yong Ho, S. Brawner, J. Lee, S. Mennicken, N. Picard, D. Schulze, and M. L. Littman, "Trigger-action programming in the wild: An analysis of 200,000 ifttt recipes," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 2016, pp. 3227–3231.

[18] —, "Owasp internet of things project - owasp," https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Attack_Surface_Areas.

[19] A. Costin, A. Zarras, and A. Francillon, "Automated dynamic firmware analysis at scale: a case study on embedded web interfaces," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. ACM, 2016, pp. 437–448.

[20] —, "Iot security foundation," https://iotsecurityfoundation.org/.

[21] ——, "https://csrc.nist.gov/csrc/media/publications/sp/800-53/rev-5/draft/documents/sp800-53r5-draft.pdf," https://csrc.nist.gov/csrc/media/publications/sp/800-53/rev-5/draft/documents/sp800-53r5-draft.pdf.

[22] L. Barreto, A. Celesti, M. Villari, M. Fazio, and A. Puliafito, "An authentication model for iot clouds," in *Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on*. IEEE, 2015, pp. 1032–1035.

[23] K. A. McKay, K. A. McKay, L. Bassham, M. S. Turan, and N. Mouha, *Report on lightweight cryptography*. US Department of Commerce, National Institute of Standards and Technology, 2017.

[24] N. Apthorpe, D. Reisman, and N. Feamster, "A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic," *arXiv preprint arXiv:1705.06805*, 2017.

[25] F. Loi, A. Sivanathan, H. H. Gharakheili, A. Radford, and V. Sivaraman, "Systematically evaluating security and privacy for consumer iot devices," in *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*. ACM, 2017, pp. 1–6.

[26] —, "Dns privacy project homepage - dns privacy project - global site," https://dnsprivacy.org/wiki/display/DP/DNS+Privacy+Project+Homepage.

[27] ——, "Smartthings shield for arduino smartthings support," https://support.smartthings.com/hc/en-us/articles/213652126-SmartThings-Shield-for-Arduino.

[28] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang, and H. Zhu, "Homonit: Monitoring smart home apps from encrypted traffic," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 1074–1088.

[29] —, "Edit distance - wikipedia," https://en.wikipedia.org/wiki/Edit_distance.

[30] N. Apthorpe, D. Y. Huang, D. Reisman, A. Narayanan, and N. Feamster, "Keeping the smart home private with smart (er) iot traffic shaping," *arXiv preprint arXiv:1812.00955*, 2018.

[31] M. Alhanahnah, Q. Lin, Q. Yan, N. Zhang, and Z. Chen, "Efficient signature generation for classifying cross-architecture iot malware," in *2018 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2018, pp. 1–9.

[32] L. S. Huang, A. Rice, E. Ellingsen, and C. Jackson, "Analyzing forged ssl certificates in the wild," in *Security and privacy (sp), 2014 ieee symposium on*. IEEE, 2014, pp. 83–97.

[33] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "Blindbox: Deep packet inspection over encrypted traffic," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 213–226, 2015.

[34] "Nokia threat intelligence report 2018," https://bit.ly/2XW34iY.

[35] J. Obermaier and M. Hutle, "Analyzing the security and privacy of cloud-based video surveillance systems," in *Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security*. ACM, 2016, pp. 22–28.

[36] "Oauth - wikipedia," https://en.wikipedia.org/wiki/OAuth.

[37] "Openid connect - wikipedia," https://en.wikipedia.org/wiki/OpenID_Connect.

[38] W. He, M. Golla, R. Padhi, J. Ofek, M. Dürmuth, E. Fernandes, and B. Ur, "Rethinking access control and authentication for the home internet of things (iot)," in *27th USENIX Security Symposium (USENIX Security 18). USENIX Association, Baltimore, MD*, 2018, pp. 255–272.

[39] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[40] Q. Zhang, Y. Wang, X. Zhang, L. Liu, X. Wu, W. Shi, and H. Zhong, "Openvdap: An open vehicular data analytics platform for cavs," in *Distributed Computing Systems (ICDCS), 2017 IEEE 38th International Conference on. IEEE*, 2018.

[41] Q. Zhang, Q. Zhang, W. Shi, and H. Zhong, "Firework: Data processing and sharing for hybrid cloud-edge analytics," *IEEE Transactions on Parallel and Distributed Systems*, 2018.