All Your IoT Devices Are Belong to Us: Security Weaknesses in IoT Management Platforms

Bhaskar Tejaswi
Concordia University
Montreal, Canada
b_tejasw@ciise.concordia.ca

Mohammad Mannan
Concordia University
Montreal, Canada
mmannan@ciise.concordia.ca

Amr Youssef
Concordia University
Montreal, Canada
youssef@ciise.concordia.ca

ABSTRACT

IoT devices have become an integral part of our day to day activities, and are also being deployed to fulfil a number of industrial, enterprise and agricultural use cases. To efficiently manage and operate these devices, the IoT ecosystem relies on several IoT management platforms. Given the security-sensitive nature of the operations performed by these platforms, analyzing them for security vulnerabilities is critical to protect the ecosystem from potential cyber threats. In this work, by exploring the core functionalities offered by leading platforms, we first design a security evaluation framework. Subsequently, we use our framework to analyze 42 IoT management platforms. Our analysis uncovers a number of high severity unauthorized access vulnerabilities in 9/42 platforms, which could lead to attacks such as remote SIM deactivation, IoT SIM overcharging and device data forgery. Furthermore, we find broken authentication in 11/42 platforms, including complete account takeover on 7/42 platforms, along with remote code execution on one of the platforms. Overall, on 11/42 platforms, we find vulnerabilities that could lead to platform-wide attacks, that affect all users and all devices connected to those platforms.

CCS CONCEPTS

• **Security and privacy → Web application security**.

KEYWORDS

IoT management platforms; IoT security; Web security

ACM Reference Format:
Bhaskar Tejaswi, Mohammad Mannan, and Amr Youssef. 2023. *All Your IoT Devices Are Belong to Us:* Security Weaknesses in IoT Management Platforms. In *Proceedings of the Thirteenth ACM Conference on Data and Application Security and Privacy (CODASPY '23), April 24–26, 2023, Charlotte, NC, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3577923.3583636

1 INTRODUCTION

IoT devices play a significant role in our daily lives (e.g., home automation), as well as at the enterprise level (e.g., device fleet management). A key component of the IoT ecosystem is an IoT platform, which hosts a number of endpoints supporting the business operations utilizing IoT devices [9]. Some of these platforms provide data management services to enable data collection from the IoT devices, followed by its processing and analytics. Some platforms offer device management services for users (e.g., enterprise IoT device administrators) to remotely connect to their devices. by using a platform's web portal and APIs. Cellular IoT SIM sellers provide connectivity management services [4], to facilitate their customers to easily and efficiently manage all of their IoT SIM cards (e.g., remotely activate/deactivate SIM cards). Such versatile functionalities, if improperly designed/implemented, can result in serious security issues (e.g., [3]).

We define an IoT management platform as a platform that provides either one or a combination of these services for IoT devices. These platforms can be used by enterprises for the devices used by them, or businesses that sell IoT devices to consumers. We design and implement a generalized security framework to evaluate the security posture of IoT management platforms from an external attacker's perspective, focused on the key services provided by them—i.e., connectivity, device, and data management. Our evaluation framework comprises a wide range of vulnerabilities such as broken authentication, unauthorized access, vulnerable trigger-action function and lack of input validation. For the evaluation, we rely on a combination of semi-automated and manual vulnerability detection techniques, that are carefully applied not to interfere with the platform operations. The scope of evaluation comprises the web requests generated upon using the platforms' websites, and those corresponding to the platforms' stand-alone APIs.

Contributions and notable findings.

(1) We design a comprehensive security evaluation framework for evaluating various complex functionalities offered in modern IoT management platforms. We include tests pertinent to core platform services—connectivity, device, and data management for operating a large number of IoT devices.
(2) We apply our framework on real-world IoT management platforms of various sizes that offer a wide range of services. For space limitations, throughout the rest of the paper, we report the analysis results for only 42 of the analyzed platforms. Our analysis uncovered vulnerabilities in 28/42 platforms; on 11/42 platforms, these lead to platform-wide attacks.
(3) The unauthorized access vulnerabilities in 9/42 IoT platforms could be abused to launch attacks such as arbitrary SIM deactivation, unauthorized Short Message Service (SMS) delivery and forged data submission from IoT devices.
(4) Broken authentication found in 11/42 platforms, with serious consequences such as full account takeover, sensitive information disclosure, and denial of service.

CODASPY '23, April 24–26, 2023, Charlotte, NC, USA

ACM ISBN 979-8-4007-0067-5/23/04…$15.00
https://doi.org/10.1145/3577923.3583636

(5) A vulnerable trigger-action function (cf. [10]) in TheThings.io grants root access to a Kubernetes container shared across platform users, by breaking out of their JavaScript sandbox.

Ethical considerations and responsible disclosure. We performed all the tests on our own accounts. For inadvertent access to sensitive data (e.g., authentication credentials in error messages), as per our university's ethics guidelines, we informed the affected platform in a timely fashion and did not retain the data. We did not perform any active scanning via automated tools for vulnerability detection and exploitation, to avoid any adverse effect on the day-to-day usage of the web-hosted platforms. We reported our findings to all the affected platforms via emails/support tickets.

2 BACKGROUND AND THREAT MODEL

In this section, we summarize key functionalities offered by IoT management platforms, and provide our threat model.

IoT connectivity management. IoT SIM cards (also called as programmable wireless SIM cards, Machine to Machine/M2M SIM cards) can typically be purchased from the providers' websites after creating a user/business account on the portal; some providers do not sell SIM cards to individual users, and involve a manual verification process. Connectivity management lets enterprise users manage the SIM card's state and connection, set data usage limits, send/receive SMS messages (e.g., for commands/outputs), generate usage reports, and set alerts for anomalous behavior (e.g., exceeding data consumption limit).

IoT device management. IoT device management services offer a centralized web portal to perform key administrative tasks on the IoT devices, as well as web APIs to enable automation. Key features in device management include [2]: provisioning and authentication of devices (devices are assigned unique IDs, and authentication tokens, which are used for onboarding on the platform); adjusting the configuration of the IoT device as needed (e.g., adding a new variable in the JSON object sent to the platform, modifying the API's URL), monitoring usage, and diagnostics for troubleshooting; and allowing enterprises to upload software/firmware updates, which are subsequently pushed to IoT devices.

IoT data management. IoT data management services enable the centralized data aggregation and processing for IoT devices. Similar to device management, the IoT device is onboarded on the platform with a unique device ID and authentication token. Data is typically submitted to the platform via protocols such as HTTP, MQTT, AMQP and COAP [11]. Users can also create visualization dashboards based on IoT device data for analytics purposes.

Threat model. The scopes of the considered attacks are defined as follows. A *platform-wide attack* affects all users and all connected devices of the platform; examples include: remote code execution via sandbox escape (platform infrastructure compromise), attacks involving broken authentication and unauthorized access issues that require the use of easily-enumerated identifier values (e.g., short numeric IDs, sequential SIM numbers). A *user-specific attack* can affect only a specific user and the devices owned by that user; examples include: session hijacking via XSS, password reset via CSRF, user credential theft via SSLStrip, attacks involving broken authentication and unauthorized access vulnerabilities that require specific user/device IDs that are not easily-enumerated/guessed (e.g., UUIDs, registered email address of the target user). A *device-specific attack* can affect a specific device (e.g., intercepting HTTP traffic to steal device authentication credentials). To perform these attacks, we assume three types of attackers in our threat model. A *user-independent remote attacker* directly interacts with the platform and does not need to involve the victim user/device in any manner. Such an attacker can create a user account on the platform's website, and perform the intended attacks. A *user-dependent remote attacker* also performs the attack remotely, but requires user involvement (such as clicking on a phishing URL). An *on-path attacker* must be on the same network path as the victim user/IoT device, and collect and analyze the traffic flow between the user's browser/IoT device and the platform's server. Attacks requiring physical access to an IoT/user device are out of scope.

3 SECURITY ANALYSIS FRAMEWORK

In this section, we provide a framework for performing the security analysis of IoT management platforms. At first, we identify key platform functionalities by manually analyzing a few platforms and perusing their API documentations. Then we select an initial list of potential security vulnerabilities that is motivated by prior research in the field of IoT security [10, 18, 20], and online services security [24]. We then iteratively refine the list of associated vulnerabilities based on their impact on the key functionalities, and focus on the vulnerabilities applicable to multiple platforms. In this section, we discuss the process of detection of each vulnerability along with the potential impact on the affected platforms, users, and connected devices. For each platform, we create two user accounts, and use one as an attacker and the other as a victim user.

3.1 Broken Authentication

Broken authentication [20] in IoT management platforms can lead to platform-wide attacks such as SIM state tampering, device data tampering, user information disclosure, arbitrary command delivery, and firmware theft. We perform the following checks to detect insecure implementation of authentication.

We log in to the platform and capture all the web requests that require authentication (e.g., cookies and API keys). We also capture requests by issuing API requests from stand-alone API collections (if provided). We exclude web requests for loading static content such as JavaScript files and images. We then remove the session credentials from each request and resend it to the platform; we label the platform as vulnerable if the response for any modified web request is the same as the original one. We use Auth Analyzer [7] for these checks, followed by manual review to assess the impact.

We check for logic bugs in the forgot password and password reset functionalities of the platforms' websites. We test if it is possible to reset the password of another user by tampering parameters (e.g., email address, username) in the underlying web requests. For platforms offering connectivity management services, we also check if the platform validates that a user owns the card they are trying to add to their account to prevent forged IoT SIM registration.

3.2 Unauthorized Access

Unauthorized access [24] can result in platform-wide attacks such as SIM state tampering, device data tampering, arbitrary SMS messages, user information disclosure, alerts configuration leakage, SMS message leakage and device data leakage. We perform the following actions to identify such issues. (a) We log in with the victim account on the platform, and capture the underlying web request to test for unauthorized access. (b) We replace the authentication details in the web request captured in step (a) with those of the attacker account, and send the modified request to the platform. (c) We observe the response of the modified request sent from the attacker's account. If the modified request is successfully processed, we flag the platform as vulnerable to unauthorized access. We use Auth Analyzer to detect unauthorized access vulnerabilities, followed by manual review of the affected requests to assess the impact. For this vulnerability, the platforms do check for the authentication credentials in the web requests, but fail to validate if the requesting users have the permission to access the resources.

3.3 Vulnerable Trigger-Action Functions

Trigger-Action Platforms (TAPs [10]) connect IoT devices and cloud services with the help of trigger-action applications. When a trigger is received by the application, certain actions are performed, which are programmed into the trigger-action application. Some IoT platforms offer similar trigger-action features on their websites, allowing users to write code for custom functionality (typically in JavaScript), e.g., unit conversion, and periodic tasks. Custom functions must be written for parsing and processing data payloads received from IoT devices (e.g., [19]).

Platforms use sandbox libraries with limited set of JavaScript methods to securely execute these user-supplied trigger-functions in an isolated environment, and to avoid attacks such as remote code execution. Popular NodeJS libraries are known to have sandbox bypass vulnerabilities and can be exploited in IoT communication (cf. [10] for TAPs vulnerabilities on IFTTT and Zapier platforms). We first try to find out the sandbox library used by the platform through a stack trace using the following JavaScript code: `function main(params, callback){callback(new Error().stack);}` If the detected JavaScript sandboxing library has known vulnerabilities, we check for those vulnerabilities, and use benign system commands such as `id` for confirming the privilege level of the system access granted by the vulnerability. We follow a process of coordinated disclosure with affected platforms to minimize any accidental system impact.

3.4 Lack of Input Validation

IoT management platforms must validate the data received from IoT devices and users (via web dashboards/APIs) before processing them. The lack of input validation [18] enables attacks, e.g., XSS and SQL injection. While XSS can lead to user-specific attacks, e.g., account takeover, SQL injection can cause platform-wide database compromise. We detect the presence of these vulnerabilities by sending web requests with malformed input parameters and analyzing the corresponding web responses. To detect XSS, we provide custom JavaScript payloads (e.g., `<script>alert(1)</script>`) in the input parameters, and check if the supplied payload is executed in the browser while navigating the website. Similarly, we checked if SQL errors are returned upon appending a single quote at the end of input parameter values. Due to ethical concerns, all instances of this vulnerability on web-hosted platforms have been solely detected via manual inspection.

4 TARGET PLATFORMS AND ANALYSIS SETUP

Target platforms. Since the IoT platforms are used mostly by enterprise users/developers and not meant for mass consumption, typical website ranking services (such as Tranco [14]) cannot be used for platform selection. Instead, we rely on a combination of the following sources to gather the list of platforms: (a) survey papers on IoT platforms [8, 22, 23]; (b) top search engine results with keywords such as IoT platform, IoT device management, IoT data management, IoT connectivity, IoT SIM, M2M SIM.

Analysis setup. We perform a black-box security assessment of the platforms and analyze their web applications and web APIs. We use Burp Suite [15] as a man-in-the-middle proxy to intercept the web requests and send crafted web requests to the platforms.

We use the following approaches in our analysis. We perform manual testing to check lack of input validation and vulnerable trigger-action functions; no automated scanning tools are used to avoid affecting the platform operations. For trigger-action functions, we manually explore the available sandbox-escape attacks for a given library. The following tests are semi-automated: broken authentication and unauthorized access (manually exploring key functionalities on the website to capture underlying web requests). To mimic the behavior of real IoT devices, we used Curl and custom Python scripts to issue test HTTP requests to the platforms. We needed to install an edge agent on an IoT device to perform the analysis on 10/42 IoT platforms. We installed Linux-compatible versions of these agents inside virtual machines.

5 RESULTS

We use our framework to analyze the security posture of 42 IoT management platforms (tested between January 2021 and June 2022). We summarize the attack types affecting various functionalities in Table 1. In this section, we discuss some of the findings of our analysis; Table 2 provides an overview of the findings based on the discovered instances of data exposure and malicious write along with the type of attacker and the scope of attack.

5.1 Broken Authentication

We found instances of broken authentication in 11/42 platforms. We provide the details below, grouped by the affected resources.

Account information. On AskSensors, an attacker can obtain sensitive account information for any user, by providing a 4-digit ID value, which can be easily enumerated for all users. Details such as username, email ID, number of connected IoT devices, account creation date, user's address and the password reset token are exposed. On Fogwing's analytics portal, an attacker could reset the password of *any* user by providing the victim's registered email ID. An attacker could view sensitive information of *any* user of Aeris Neo, by providing a 5-digit account ID of the victim. Using a trial

Table 1: Overview of the discovered attacks, their scope, and corresponding vulnerable platform functionalities (denoted by ✗)

Attack Scope	Attack Description	Vulnerable Platform Functionalities							
		SIM Mgmt.	Alerts Mgmt.	SMS	IMEI Lock	Device Commands	Firmware Updates	Device Data Handling	User Accounts
Platform-wide	Sandbox escape		✗						
	Arbitrary command issuance					✗			
	Mistimed alerts		✗						
	SIM state tampering	✗			✗				
	Device data tampering							✗	
	Arbitrary SMS messages			✗					
	User information disclosure								✗
	Alerts config. leakage		✗						
	SMS message leakage			✗					
	Firmware theft						✗		
	Device data leakage					✗		✗	
User-specific	Account takeover								✗
	Partial account modification								✗
Device-specific	API key theft					✗	✗	✗	

Table 2: Overview of the discovered instances of data exposure and malicious write along with type of attacker and scope of attack (see Sec. 2). Any remote attacker can perform platform-wide (●) or user-specific (⊘) attacks; a user-dependent remote attacker can perform platform-wide (◉) or user-specific (○) attacks; any on-path attacker can perform user-specific (⊗) or device-specific (⊙) attacks. In instances exploitable by multiple attacker types, we consider the worst one (e.g., remote attackers are worse than on-path attackers), with the broadest scope.

	Data Exposure					Malicious Write									
	Alerts Config.	SMS Messages	Device Metadata	Command output	Account Info	SIM State	SIM Registration	Alerts Config.	SMS Messages	IMEI Lock	Device Commands	Forged IoT data	Dashboard	Account Takeover	DB Modification	Potential RCE
Verizon's Thingspace													○			
Aeris Neo	●	●			●	●		●	●					●		
RemoteIOT													⊘			
OneSIMCard		●				●	●		●	●				○		
Hologram							●									
KeepGo	●	●				●	●	●	●							
Tago			⊙									⊙	⊙			
Favoriot	●		●		●			●				●	●	●		
TheThings.io	●				●			●				○	●	○		●
Mdash	○		○	○	○			○			○	○	○	○		
Fogwing	●		●		●			⊘				⊘	⊘	⊘		
Asksensors	●		●		●			●			●	●	●	●		
CSL	⊗				⊗	⊗		⊗						⊗		
GlobalM2MSIM	⊘		⊘		⊘	⊘		⊘	⊘				⊘	⊘		
Imvvy			⊗		⊗							⊗	⊗	⊗		
Open M2M		●			○	○			○					○		
ResIOT	⊘	⊘	⊘		●	⊘		⊘	⊘				⊘	⊘		
Thingsboard					○									○		

and error approach, the attacker can retrieve sensitive information, e.g., name, email ID, account type and API key for other users. In ResIOT, an attacker could get any user's authentication token by providing the victim's email address. Thereafter, the attacker could log in to the victim user's account on the platform, and perform IoT SIM management tasks.

IoT device. Broken authentication could be abused for remote command execution and device data forgery. An attacker could send arbitrary commands to the IoT devices connected to AskSensors, where the attacker needs to supply the command and the device ID (a 5-digit numeric value) in a POST request. An attacker could also obtain the GPS coordinates of each IoT device on the platform. We found broken authentication on Favoriot that could let an attacker obtain sensitive information such as user ID, Bcrypt hashed password, API keys with read-only and read-write permissions—by providing only the email address of a victim user.

IoT SIM registration. We found a lack of authentication in the IoT SIM card registration process of KeepGo, Hologram and OneSIMCard. KeepGo users can register their IoT SIM card on the platform by entering their Integrated Circuit Card Identification (ICCID) number; no other verification is required. Upon registration, a request is sent to the platform to check the SIM card's availability. We

altered the last few digits in our own IoT SIM card's ICCID number and found an unassigned ICCID number within under 100 requests, which we could successfully register from another test account. Similarly, an attacker can enumerate valid unassigned ICCID (in Hologram) and SIM numbers (in OneSIMCard) and add them to fake accounts. If such an unassigned SIM card is later purchased by a customer, she would receive an error message during registration of the SIM card (and may require intervention of the support team).

5.2 Unauthorized Access

We found that 9/42 platforms are vulnerable to unauthorized access. We provide the details below.

IoT SIM. On KeepGo, users can create sub accounts within their own account and assign IoT SIM cards to these sub accounts. An attacker can view other users' sub accounts, by inputting a 4-digit account ID (easily enumerated). More importantly, an attacker can deactivate another user's sub account, which deactivates all the IoT SIM cards assigned to that sub account; the attacker needs to send the victim sub account's ID. Such an attack can be performed by anyone with a registered account on the platform, with or without an IoT SIM card from KeepGo. On OneSIMCard, an attacker can block the cellular connectivity and internet access for *any* IoT SIM in the platform, by sending web requests with the SIM number of the victim, which is a unique 15-digit numeric value.[1] Only existing users with SIM cards assigned to their accounts can perform these attacks. An attacker can use the broken authentication in SIM registration process (see Sec. 5.1) to obtain such an account.

Alerts. On KeepGo, users can set rules to trigger an email alert notification if the account balance falls under a set threshold. An attacker could modify this rule for any user by providing account ID (a 4-digit number), condition value (containing the threshold amount in USD) and the desired email address in a POST request. The attacker could set a large negative value as the threshold and as a result, the alert would not be triggered. An attacker could also redirect the alerts to an arbitrary email address. In both cases, the user would not receive timely notifications, which may lead to service disruption.

TheThings.io contains a module named *Cloud Code*, allowing users to write jobs, functions, and triggers on the platform [19]. An attacker can view the cloud codes of other users, by providing a 5-digit organization ID (easily enumerated). TheThings.io lets users utilize third party services e.g., Twilio (for SMS/voice alerts), and SendGrid, Mandrill, SES, and Gmail (for emails) while defining triggers. The unauthorized access vulnerability in *Cloud Code* exposes the authentication credentials (e.g., API keys, tokens) for these third party services as well. These credentials can be abused in several ways, e.g., to retrieve sensitive information such as metadata of emails previously sent, to use the service APIs for free (incurred cost will be billed to the victim), and to launch phishing attacks via emails and SMS messages.

SMS. KeepGo users can send SMS messages to their IoT SIM cards from the platform, and the responses from the IoT devices can be viewed on KeepGo's website. For each inbound and outbound message, the user is charged 0.05 USD. An attacker could view all the exchanged IoT SMS messages by providing the ICCID of the victim's IoT SIM card (see Sec. 5.1, under "IoT SIM Registration"), and the billing cycle. More importantly, an attacker can send arbitrary messages by inputting the target SIM's ICCID and the SMS text in a POST request, after which, the victim's account balance is reduced by 0.05 USD. Similarly, on OneSIMCard, an attacker could view messages sent to any SIM card, and send arbitrary messages to an IoT SIM card by inputting the target SIM number and the SMS text in a POST request, for which the victim's account balance is reduced by 0.01 USD. Note that the attacker's account is not charged at all in these attacks.

IoT device data. On Fogwing, an attacker could view sensitive information of any IoT device by providing the 4-digit gateway ID; exposed details include name, edge ID, geolocation, health status, and data received from the device. An attacker could further abuse the leaked edge ID to send forged data on behalf of the targeted IoT device to the platform by providing the victim's edge ID and the attacker's API key as URL parameters along with the forged data payload in the POST request. On AskSensors, for any device, simply by providing a 4-digit ID, an attacker could view the API keys, and remove the device. An attacker could also read data submitted by any IoT device, and send forged data on behalf of any device to the platform. The exposed API key could be further abused to launch XSS attacks (see Sec. 5.4).

5.3 Vulnerable Trigger-Action Functions

On TheThings.io, users can run JavaScript code in the Cloud Code module on the platform. These code segments are run in a sandbox [19]. From an error stack, we inferred that the platform uses Jailed sandbox. We followed a coordinated disclosure approach, wherein we were provided a perpetual access account by the platform's CEO for further testing. We were able to bypass the sandbox using known sandbox bypass attacks and execute system commands with root privileges using the child_process and process.exec() modules (from Node.js). We found that the Cloud Codes functionality was running inside a shared kubernetes pod and were able to gain root access on it. No further actions were taken that could alter system configuration or ex-filtrate sensitive information.

5.4 Lack of Input Validation

We found cross-site scripting (XSS) vulnerabilities on 12/42 evaluated platforms. On 6 of them (OpenM2M, OneSIMCard, Favoriot, TheThings.io, AskSensors, Thingsboard), an attacker could steal session cookies and authentication tokens stored in browser's LocalStorage/SessionStorage from active user sessions via XSS. Note that on each of these 6 platforms, we attained stored XSS, providing an attacker perpetual access to session cookies/tokens whenever the user visits the affected pages, leading to account takeover. On KeepGo, by exploiting the unauthorized access vulnerability in the SMS functionality (see Sec. 5.2), an attacker can send a JavaScript payload as an SMS message, which executes when the IoT SIM owner views the list of sent SMS messages on the platform's web portal. On TheThings.io and Imvvy, an on-path attacker could capture the authentication credentials from HTTP requests and MQTT(without TLS) messages, respectively and abuse them to launch XSS attacks.

[1] As from the purchased cards, OneSIMCard apparently uses sequential SIM numbers.

6 LIMITATIONS

For 14/20 platforms offering connectivity management services, and for all the platforms offering device and/or data management services, we performed the assessment with trial accounts with a limited set of functionalities. Thus our findings may represent a lower bound of the vulnerabilities. Several platforms do not allow self-registration of user accounts, and require manual verification (e.g., proof of business ownership) before granting access to the platform; we excluded such platforms. Also, most of the vulnerabilities tested and their detection techniques are no different from those adopted in traditional web security. However, results from our vulnerability assessment demonstrate the practical consequences of such known issues in the IoT management platforms.

7 RELATED WORK

Here, we summarize related past research in IoT cellular connectivity, and IoT security in general.

IoT cellular connectivity. Trend Micro, in collaboration with Europol [6], studied how IoT SIM cards from compromised IoT devices are misused for committing cyber telecom frauds such as subscription fraud (i.e., abusing business processes to access sensitive data from victim's account), and toll fraud (i.e., initiating high volumes of expensive international calls). Some cellular connectivity providers offer lower data charges for IoT SIMs compared to non-IoT SIM cards. Past research [21] has revealed that it is possible to use IoT SIM cards of such providers outside the IoT devices (e.g., in a smartphone), causing financial loss to the connectivity providers. In a recent BlackHat presentation [17], a study on 9 IoT platforms found vulnerabilities such as unauthorized access, insecure communication and XSS.

IoT security. Past research mostly focused on specific domains—e.g., home-automation [1], video-surveillance [12], smart buildings [16]. They relied on vendor-specific devices and mobile companion applications, and as such, did not cover all the platform APIs comprehensively. Our work is device/app-agnostic, and covers both client-side and management APIs—the latter APIs are not covered in prior work. We also cover websites and stand-alone APIs from a variety of IoT platforms: consumer IoT (e.g., Tuya), enterprise IoT (e.g., thethings.io, Kaa IoT), and industrial IoT (e.g., Siemen's Mindsphere, Fogwing), and more generic IoT platforms (e.g., AWS, Azure). Past work on trigger-action platforms [10] motivated us to check for sandbox escape issues in trigger-action functionalities on IoT platforms. Insecure ecosystem interfaces are included in OWASP's list of top 10 security issues in IoT systems [13]. The websites and web APIs of IoT platforms are among the most vital interfaces and vulnerabilities in them can be abused to target a large set of users and their devices. IoT platforms have been compared based on the security features mentioned in their documentations [5]; however, no actual security evaluation was performed.

8 CONCLUSION

We provide a security evaluation framework for IoT management platforms which offer data management, device management and connectivity management services for consumer/business/industrial IoT devices. We use our framework to perform a systematic review of real world platforms. Our security analysis revealed major unauthorized access flaws in 9 platforms. We also uncovered other severe vulnerabilities such as broken authentication in 11 platforms, and remote code execution on one of the evaluated platforms. We hope that our study would help developers secure their platforms against these easy-to-launch but severe attacks.

ACKNOWLEDGMENTS

We are grateful to the anonymous CODASPY 2023 reviewers for their insightful suggestions. This work is partially supported by Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

[1] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. Sok: Security evaluation of home-based IoT deployments. In *IEEE Symposium on Security and Privacy*. San Francisco, CA, USA.
[2] Avsystem.com. 2020. *IoT device management: definition and fundamentals.* Online article (Aug 28, 2020). https://www.avsystem.com/blog/iot-device-management/.
[3] Duo.com. 2021. *Critical bug in Kalay IoT protocol threatens millions of devices.* Online article (Aug 18, 2021). https://duo.com/decipher/critical-bug-in-kalay-iot-protocol-threatens-millions-of-devices.
[4] Emnify.com. 2020. *What is a Connectivity Management Platform (CMP)?* Online article (Dec 10, 2020). https://www.emnify.com/iot-glossary/connectivity-management-platform.
[5] G. Fortino, A. Guerrieri, P. Pace, C. Savaglio, and G. Spezzano. 2022. IoT Platforms and Security: An Analysis of the Leading Industrial/Commercial Solutions. *Sensors* 22, 6 (March 2022).
[6] Gibson, Craig. 2018. Toll Fraud, International Revenue Share Fraud and More: How Criminals Monetise Hacked Cellphones and IoT Devices for Telecom Fraud.
[7] Github.com. [n. d.]. Auth Analyzer. https://github.com/portswigger/auth-analyzer
[8] Hamdan Hejazi, Husam Rajab, Tibor Cinkler, and László Lengyel. 2018. Survey of platforms for massive IoT. In *IEEE International Conference on Future IoT Technologies*. Eger, Hungary.
[9] Gartner Inc. [n. d.]. Definition of IoT Platforms. https://www.gartner.com/en/information-technology/glossary/iot-platforms.
[10] Mohammad M. Ahmadpanah and Daniel Hedin and Musard Balliu and Lars Eric Olsson and Andrei Sabelfeld. 2021. SandTrap: Securing JavaScript-driven Trigger-Action Platforms. In *USENIX Security Symposium*. Online.
[11] Nitin Naik. 2017. Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In *IEEE ISSE'17*. IEEE, 1–7.
[12] J. Obermaier and M. Hutle. 2016. Analyzing the security and privacy of cloud-based video surveillance systems. In *ACM IoTPTS'16*. Xi'an, China.
[13] Owasp.org. [n. d.]. OWASP Internet of Things Project. https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10.
[14] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. [n. d.]. Tranco. https://tranco-list.eu.
[15] Portswigger.net. [n. d.]. Burp Suite. https://portswigger.net/burp.
[16] L. Rondon, L. Babun, A. Aris, K. Akkaya, and A. Uluagac. 2021. LightningStrike: (in)secure practices of E-IoT systems in the wild. In *ACM WiSec'21*. Abu Dhabi, United Arab Emirates.
[17] Altaf Shaik and Shinjo Park. 2022. Attacks from a New Front Door in 4G & 5G mobile networks. https://i.blackhat.com/USA-22/Wednesday/US-22-Shaik-Attacks-From-a-New-Front-Door-in-4G-5G-Mobile-Networks.pdf.
[18] Mark Stanislav and Tod Beardsley. 2015. Hacking IoT: A case study on baby monitor exposures and vulnerabilities. (2015). Online article (Sept, 2015). https://www.rapid7.com/globalassets/external/docs/Hacking-IoT-A-Case-Study-on-Baby-Monitor-Exposures-and-Vulnerabilities.pdf.
[19] TheThings.io. [n. d.]. Cloud Code Sandbox. https://developers.thethings.io/docs/cloud-code-sandbox.
[20] Xueqiang Wang, Yuqiong Sun, Susanta Nanda, and XiaoFeng Wang. 2019. Looking from the mirror: Evaluating IoT device security through mobile companion apps. In *USENIX Security Symposium*. Santa Clara, CA, USA.
[21] Tian Xie, Guan-Hua Tu, Chi-Yu Li, and Chunyi Peng. 2020. How can IoT services pose new security threats in operational cellular networks? *IEEE Transactions on Mobile Computing* 20, 8 (2020), 2592–2606.
[22] J. Yu and Y. Kim. 2019. Analysis of IoT platform security: A survey. In *International Conference on Platform Technology and Service*. Jeju, South Korea.
[23] M. Zdravković, M. Trajanović, J. Sarraipa, R. Jardim-Gonçalves, M. Lezoche, A. Aubry, and H. Panetto. 2016. Survey of Internet-of-Things platforms. In *International Conference on Information Society and Techology*. Barcelona, Spain.
[24] C. Zuo, Q. Zhao, and Z. Lin. 2017. Authscope: Towards automatic discovery of vulnerable authorizations in online services. In *ACM CCS'17*. Dallas, TX, USA.