TRUST MODELLING FOR SECURITY OF IOT DEVICES

Naresh K. Sehgal[1], Shiv Shankar[2] and John M. Acken[3]

[1]Data Centre Group, Intel Corp, Santa Clara, CA
[2]Chief Data Scientist, Maphalli, Bangalore, India
[3]ECE Department, Portland State University, Portland, OR

ABSTRACT

IoT (Internet of Things), represents many kinds of devices in the field, connected to data-centers via various networks, submitting data, and allow themselves to be controlled. Connected cameras, TV, media players, access control systems, and wireless sensors are becoming pervasive. Their applications include Retail Solutions, Home, Transportation and Automotive, Industrial and Energy etc. This growth also represents security threat, as several hacker attacks been launched using these devices as agents. We explore the current environment and propose a quantitative and qualitative trust model, using a multi-dimensional exploration space, based on the hardware and software stack. This can be extended to any combination of IoT devices, and dynamically updated as the type of applications, deployment environment or any ingredients change.

KEYWORDS

Edge Computing, Security, Adaptive learning, Trust model, Threats, Cloud Computing, Information Security

1. INTRODUCTION

Security concerns [1] abound with the emergence of IoT devices in Cloud Computing. A recent DDOS (Distributed Denial of Service) attack was launched using hijacked home security cameras, while in another instance private video clips were stolen and posted on Internet. Vulnerabilities in other unprotected devices, such as home appliances (TV, Fridge) on a network can be used to launch a cyber attack.

IoT devices are constantly collecting data about an environment or individuals, which can be potentially shared with third parties compromising privacy. It can range from personal preferences of web-browsing habits, TV channels selection, or images from home security cameras. In addition, there are security concerns if access controls to these IoT devices are compromised. An example is of someone hacking into a home control system to open garage doors or alter air-conditioning settings. While the latter may represent a minor inconvenience for a homeowner, if done for many homes at once can result in an overload of the local electric grid. Furthermore, if these devices connect to a service provider then its servers can be accessed via the devices to compromise its security. If IoT devices are located in a factory then an unauthorized access can be used to harm the equipment or products being manufactured. If these IoT devices are deployed in a hospital, then patient care can be compromised. At an individual level, it may mean incorrect readings from a blood sugar monitor resulting in inappropriate dosage of insulin, potentially with fatal consequences.

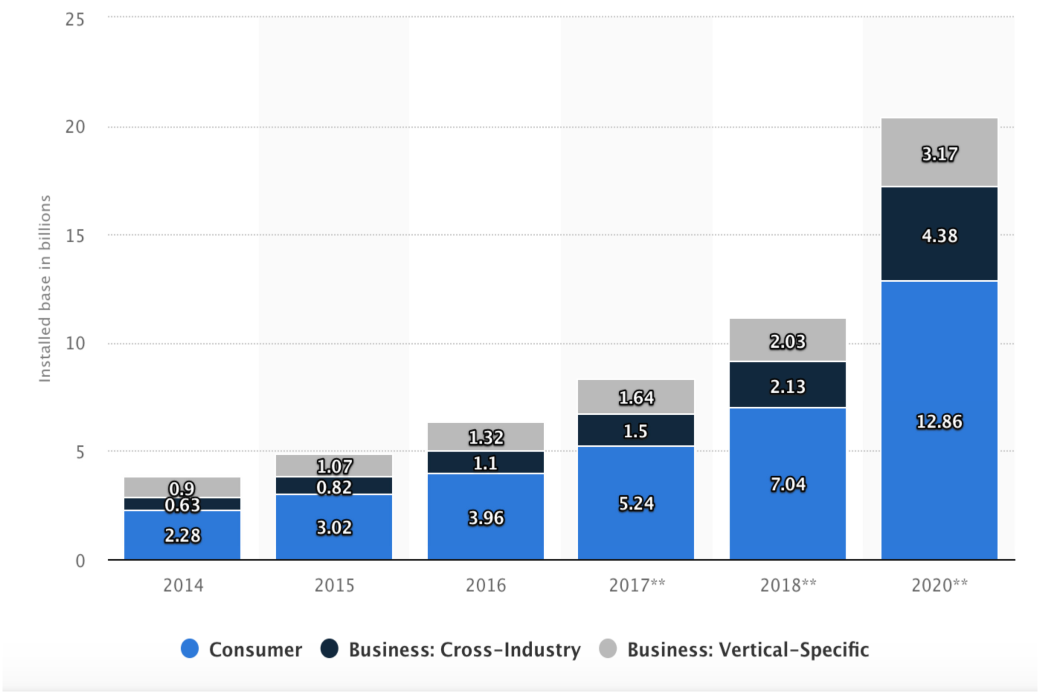


Fig 1: Growth in IoT devices over the years [2]

2. BACKGROUND

Another emerging trend is a Cloud driven by things vs. current Cloud Computing mostly driven by people, as cameras and wireless sensors are becoming pervasive [2]. Growth of IOT devices, and distribution between consumers and business are shown in Figure 1. Their applications include Retail Solutions, Home, Transportation and Automotive, Industrial and Energy etc. An example of retail industry is Amazon’s user-facing portals where customers can visualize things and transact them. An example of Transportation and Automotive is a Software Defined Cockpit in a commercial aircraft, or an autonomous vehicle. An example of manufacturing is a smart factory with robots or energy savings in a building. Lastly, additional market segments such as health, print imaging, gaming and education are being digitized at an unprecedented rate. The phrase “Internet of things” was first used by British technology visionary Kevin Aston in 1999. His perception was to think of “objects in physical world connected by sensors”. Internet Architecture Board (IAB) RFC 7452 provides the definition of IoT, as follows:

"Internet of Things" (IoT) denotes a trend where a large number of embedded devices employ communication services offered by Internet protocols. Many of these devices, often called "smart objects,’’ are not directly operated by humans, but exist as components in buildings or vehicles, or are spread out in the environment. Four basic communication models for IoT are:

1. Device to device
2. Device to cloud
3. Device to gateway
4. Backend data sharing model

We are more interested in #2 and #4, as both involve Cloud services. An example is shown in figure 2, of home appliances such as a thermostat controlled A/C connected to Cloud for better energy management [3]

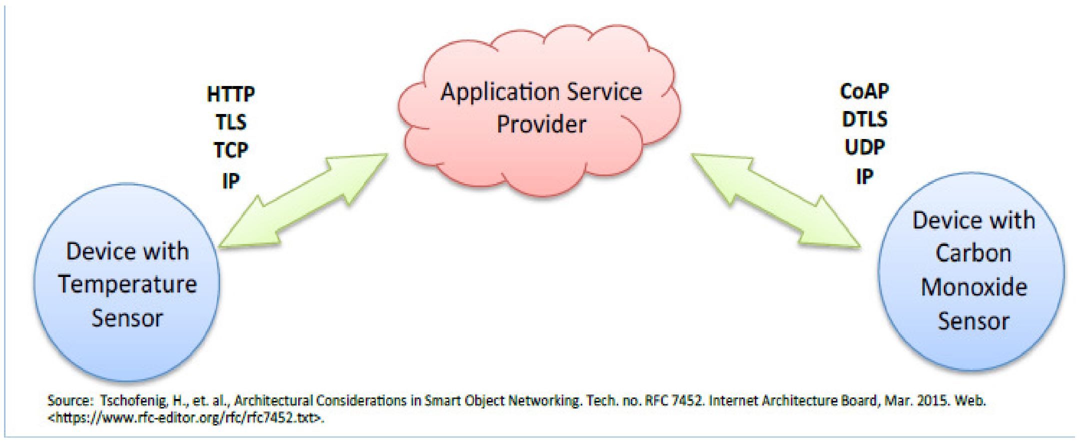

HTTP
TLS
TCP
IP
Application Service Provider
CoAP
DTLS
UDP
IP
Device with Temperature Sensor
Device with Carbon Monoxide Sensor
Source: Tschofenig, H., et. al., Architectural Considerations in Smart Object Networking. Tech. no. RFC 7452. Internet Architecture Board, Mar. 2015. Web. <https://www.rfc-editor.org/rfc/rfc7452.txt>.

Fig 2: Cloud based energy management, monitoring and optimization [3]

3. Security Attacks Using IoT Devices

For ensuring trust in IoT based Cloud Computing, it has to start with a trusted environment, trusted protocols and tamper proof components. Vendors need to provide "anti-tamper" solutions. Software upgrades in the field are needed for any bug fixes during the lifetime of an IoT device. A secure channel must exist to provide signed data that are transmitted and installed in the field, e.g., on a car or TV at home. In our door example, the vendor needs to provide an anti-tamper solution, to prevent someone locally changing the firmware or settings in an unauthorized manner. Even remote software upgrades are authenticated, as unprotected home appliances can be used to launch cyber attacks, e.g., someone using a collection of botnets to launch a DDOS attack on a Cloud server, where a botnet refers to one or more IoT devices being remotely controlled like a robotic army. Besides security, there are privacy concerns, as home sensors are collecting data about individuals that can be shared with third parties for commercial and political purposes.

Undesirable consequence may emerge if a third party can remotely gain control, e.g., of a self-driven car causing an accident on the road, or someone with malice can access the medicine drip-meters in a hospital with fatal consequences for the patients. This can be avoided with a balanced approach to interoperability and access control. This needs to be addressed at different layers of architecture, and within the protocol stacks between the devices. Standardization and adoption of communication protocols should specify when it is optimal to have standards. Some vendors like to create a proprietary ecosystem of compatible IoT products. This creates user lock-in to their particular ecosystem, which from a vendor's point of view is desirable because a closed ecosystem approach can offer benefits of security and reduces costs. However, from a user's point of view, such practices can create interoperability problems with solutions from other vendors, thereby limiting user's choices in case of upgrades or future system expansion.

As the frontiers of Cloud computing are expanding beyond the walls of a datacenter to the extremes of a network, a new term called Edge Computing is emerging. It refers to the data analytics occurring at the sources of data generation. This is bringing forth both new and existing security challenges, Following classifications describe the types of security issues as related to the Edge Computing, with a few examples:

1) **Identity authentication:** By definition, the number of players in Edge Computing is large and these may not belong to the same organization. It is infeasible to verify their

identity in a foolproof manner. Trust needs to be extended, as new customers buy their devices, such as security cameras, and bring these online with a remote registration. Central authority then must depend on the ability of these remote customers to protect their own devices.

2) **Unauthorized access:** Depending on the nature of devices at the Edge, their access into data-center may be bi-directional in nature. If someone hacks into a trusted remote device, and retrieves its authentication certificates to configure their own devices, then it will be nearly impossible to differentiate between genuine or fake users. Similarly, someone pretending to act as a central computer can access the remote devices and get critical user-data, such as on remote medical devices.

3) **Denial of service attacks:** An attack launched by hijacking multiple remote devices and simultaneously contacting the central server. This will cause the server to be overloaded, denying access to genuine users in a timely manner.

4) **Data theft:** Depending on where data is stored and for how long opens the possibility of it being stolen. An example is a security camera at home with local storage. In event of a theft, it may be possible for an intruder to simply remove the local storage, thus circumventing the purpose of a security camera. However, if camera immediately uploads an image to Cloud upon detecting a motion, then any physical tampering will not alter the images of intruders.

5) **Data integrity and falsification:** A key difference between confidentiality and integrity is that in the latter case, an attacker doesn't need to read the protected data, but merely modify it, e.g., with a buffer overflow, rendering it useless. This system level attack can happen if multiple devices from different sources are writing back to a central server database.

6) **Invasion of privacy:** Since multiple players may combine their data inputs from different sources to arrive at a desired conclusion, e.g., for real-time traffic updates, their identities need to be protected. This may include an individual's location, movements and any other aspects of personal nature.

7) **Activity monitoring:** A cell phone that constantly pings the signal tower, is sufficient for someone to monitor the location of aphone's owner, their movements etc. Furthermore, if a remote app can turn on the microphone or camera in a phone, then additional information and activities can be monitored in an illegal manner. Similar effects can be achieved with fixed cameras at commercial or public locations, e.g., in a shopping center.

8) **Rooting of devices**: Additional software can be installed in the IoT devices without users' permission. The software can 'root' the device preventing detection and have full access. There is no universal virus or malware scanner for IoT.

Some devices can be programmed to selectively transmit data to a cloud service for processing, e.g., a security camera which has a buffer of 15 seconds, but records and transmits a 30 seconds of clip only if any motion is detected, for 15 seconds before and 15 seconds after the motion is detected. This reduces storage requirements but increases chances of a missed detection. Such devices are designed to render service with minimal intervention, and yet they need to be directed using voice activation or image recognition.

These and other devices can be used to conduct a DDOS attack on the backend server, even in a serverless architecture [4]. The attacker simply hijacks one or more devices, and uses them to inundate the backend services. This can be done by sending more data, and more often, from the camera even when there is no motion detected.

4. SECURITY SOLUTIONS FOR IOT DEVICES

Solution level cost considerations involve technical factors such as limited internal processing, memory resources or power consumption demands. Vendors try to reduce the unit cost of devices by minimizing parts and product design costs. It is more expensive to design interoperability features into a product and test for compliance with a standards specification. A non-interoperable device may lack in standards and the documented best practices. It may limit the potential use of IoT device, and absence of these standards can result in deviant behavior by IoT devices.

It is recognized that traditional Trusted Compute Boundary (TCB) expands with Edge Computing to include domains that are physically outside the control of remote device or central data-center owners. The best they can do is to monitor/track a threat, identify an attacker, launch a recovery and prevent false positives. These steps are outlined below:

1) **Monitor/track a Threat:** This is possible by establishing a normal usage pattern for the IoTdevice, an example is a security camera at home, which uploads data whenever any motion is detected, e.g., whenever people go in and out. If the regular pattern for a home is no more than a couple of dozen data uploads during a day, then hundreds of data loads to the central server within a few minutes may indicate that the device has been compromised. It could be an attempt to cause a DOS attack.

2) **Identifying attackers:** Once a threat is detected, then attackers need to be identified. These could take the form of an IP address of the IoT that is repeatedly pinging the central server, to launch a denial of service attack.

3) **Attack recovery:** This can take the form of blocking the offending IP address. However, an attacker can corrupt the critical data before the attacker's presence is detected. In such a case, frequent checkpoints must be taken to do a rollback to the known good state.

4) **Accidental and unintentional failures confused with security attacks:** Any detection method suffers from the risks of false positives, e.g., mistaken flagging of genuine access as a potential threat. An example of this is a stock market trading computer that detects unusual activity, which is genuine yet may flag a false alarm. Similar situation can happen with security alarms due to false sensor activity data etc. This calls for a learning system that becomes smarter over time.

5) **Data Integrity Protection:** We previously described a system level attack if multiple devices from different sources are writing back to a central server database. This can be protected by assigning a virtual partition or container to the data coming from each distinct source, and checking the address range of each access to prevent data integrity of other users on the same server.

Internet Engineering Task Force (IETF) has identified the problem of Interoperability, as many suppliers build "walled gardens" that limit users to interoperate with a curated subset of component providers, applications and services.

Interoperability solutions between IoT devices and backend systems can exist at different layers of the architecture, and at different levels within protocol stack between the devices. Key is the standardization and adoption of protocols, which should specify when and where it is optimal to use standards. More work is needed to ensure interoperability within the cost constraints for Edge Computing to become pervasive.

There are other regulatory and policy issues at play, such as device data being collected and stored in a Cloud may cross-jurisdictional boundaries, raising liability issues if the data leaks. This is especially important if data is of personal nature, e.g., related to shopping patterns or patient health records.

5. TRUST MODELS FOR IOT DEVICES

Attacks have been made exploiting a component level vulnerability. Most security systems are designed using Capability models. A capability model usually takes into account how various services are utilized. For example, we can start with a multi-dimensional structure, composed of:

1) Hardware: An ASIC or programmable microcontroller.
2) Operating System: Windows, Linux, Android etc.
3) Applications: nature of application, and its privilege level.
4) Manner in which various components, services and utilities are deployed:

    a) e.g., kernel, library services, files accesses,
    b) Manner in which objects (username, application, function) get authenticated,
    c) What kind of cryptography is utilized, e.g., strength of MD5 vs. SHA256.

We propose to evaluate a given HW and SW solution components composed of one or more IOT devices connected to a Cloud server, based on the robustness and trustworthiness of this entire solution stack, with a multiplicative serialized model, e.g., in the following order:

1. Native compiled code is trusted more
2. Then anything using an external library
3. Lastly, any third party SW attempting to integrate

Using the above method, it is possible for us to evaluate trust of different operating systems with applications from diverse fields. Goal is to create a framework for evaluating and assigning a security score to each layer and then compute a composite score. A given application can be disassembled to see whether it is using a kernel service, or a utility in the user-space, or a built-in-library etc.

For each component in the stack, a list of orthogonal properties are established followed by an objective scoring system for each property. Numerical score for a utility function depends on the manner in which it is accessed, e.g., read (as a call by value), or a write (call by reference). A Security Score can computed by answering a set of questions by a user or automatically computed by a testing tool. Example of questions include:

- Whether a salt is used hash passwords?
- Which algorithm is used for hashing: MD5 or SHA256?
- Does the communication channel use SSL and which version of TLS is being used?
- What is the version of MYSQL in operation?

Another Security Score determination method: Whether port 3306 used by MySQL is open to the world or just to the application servers that use the MySQL database. This score can be continuously updated during the operations. More importantly, it needs to be updated after a maintenance or upgrade action is completed.

Security Score questionnaire may focus on the best practices during development. Automated score calculation focuses on the system operations. An OS without the latest patch can be at a security risk.

Security Score computations has two outputs:

1. **Probability of a successful attack:** What is the probability that an attack on this device will succeed?
2. **Probable Impact of a successful attack:** What is the probable impact if the attack succeeds?

The Security Score (S) can be computed as follows:
S = 1 - Pa * Pi
Where:
Pa - Probability of the attack in the range 0 to 1
Pi - Probable impact if the attack succeeds in the range 0 to 1
Pa * Pi - is the expected loss

This score is for a single component. By describing the security-wise relationship among the different components and their individual security scores, the whole system security score can be computed.

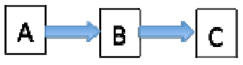
The factors that affect the probability of attack include:

- Presence of a vulnerability existing and known to attackers
- Level of focus on products of this type by hackers
- History of exploitation of this product type

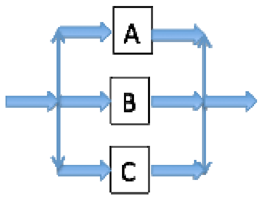
Probable impact of a security attack is defined as the sum of any regulatory fines, reputational damage and operational loss. This represents the resulting loss of trust in product and services. This needs constant monitoring for security breaches and policy updating [5].

The first step in the modelling is describe the whole system in terms of its components hierarchically organized and security-wise connections between the components could be serial or parallel.

The direction lines in figure 3 represent the security-wise relationship between the blocks in a system. In figure 3(a), to all the blocks should be secure for system to be secure and provide the required functionality. In figure 3(b), any one of the blocks should be secure for the system to provide the required functionality. The composite Security Score can be computed by applying the series-parallel reliability rules [6], as shown in Figure 3.



$$S_S = S_A * S_B * S_C$$



$$S_S = 1 - (1-S_A)*(1-S_B)*(1-S_C)$$

Figure 3: Risk levels of series-parallel systems

6. An Example Study Of Trust Scoring

Raspberry PI is the de facto choice and starting point for many IoT devices. This choice is driven by its ubiquity and low price, making it a popular controller for many home and entry level appliances. An higher installed base also makes it an attractive target for hackers, therefore we evaluated it for our IoT trust model. For our sample system, we restricted probability values to High (0.9), Medium (0.6) and Low (0.3). Similarly, the impact values were also High (0.9), Medium (0.6) and Low (0.3).

We took an implementation of a Raspberry Pi Model 3B with Raspbian OS Ver 4.14 released on 2018-4-18 as a reference system for trust scoring [7]. The base Raspberry Pi system comes with a microSD card, which holds the OS and can be used to install additional software. The factory settings and factory shipped software packages for the OS were used for trust scoring. No packages were updated. Once the basic model trust scoring was complete, we proceeded to complete the Raspberry Pi based Security Camera setup [8]. Following additional software components were installed, as depicted in Figure 4:

1. MongoDB
2. Rabbit MQ
3. AWS IOT client
4. MotionPie software

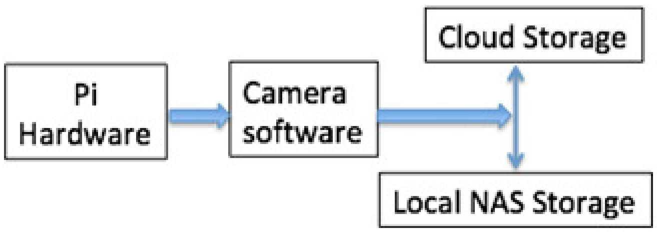


Figure 4: Series-parallel implementation of our Prototype

We use Mongo DB to have a NAS (Network Attached storage) of images, and the AWS IoT client to connect with Amazon's backend service for cloud storage. MotionPie is an image processing software to detect motion, and then decide which video clips need to be saved or discarded.

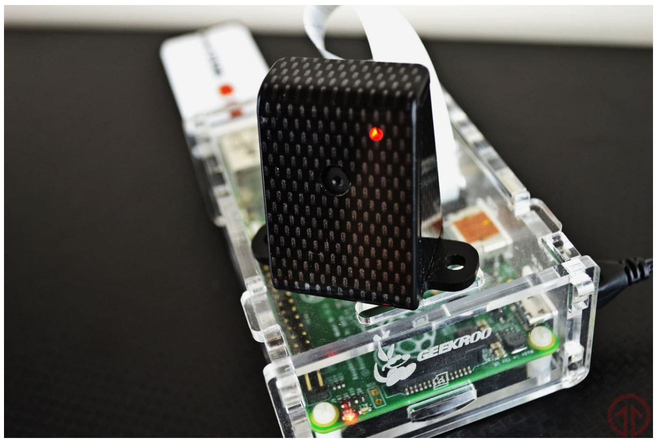


Fig 3: A simple Raspberry Pi based Camera system [6]

A problem with this security camera prototype is that someone with a physical access to local system can easily switch the software. There is no method to check if the system software is authenticated at boot time, so the base hardware setup has a high probability (0.9) of an attack. The impact probability of such attack is also high (0.9) as the base system can be fully compromised.

In the default setup, the user name is “admin”, and password is blank. It is easy for someone to remotely hijack and use this camera in a Mirai botnet attack [9]. After the password has been changed, and if the camera is installed behind a secure firewall, the probability of such an attack is medium (0.6). However, the impact probability is high (0.9). Our proposed system uses AWS IOT security model [10], with X.509 certification with asymmetric keys [8]. On the backend, where the images are stored, the security is high so probability of an attack is low (0.3) and impact probability is also low (0.3), since the images have a local storage as we as well cloud based storage. Even though the local system can be attacked with a higher probability (0.9) and medium impact (0.6).

Overall, we have the Raspberry hardware and software components in series security-wise, which itself is in series with two parallel storage systems security-wise. At component level, here is what we have so far:

$Sc = 1 - 0.9 * 0.9 = 1 - 0.81 = 0.19$
$Ss = 1 - 0.6 * 0.9 = 1 - 0.54 = 0.46$
and for the storage systems,
$Sgc = 1 - 0.3 * 0.3 = 0.91$
$Sgl = 1 - 0.9 * 0.6 = 0.46$

Where, Sc is the security of camera, Ssis the security of software, Sgc is the security of cloud storage, and Sgl is the security of local storage. As cloud storage and local storage are in parallel providing redundant functionality, the security score can be computed using reliability parallel chaining rule:

$$\begin{aligned} Sg &= 1 - (1 - Sgc) * (1- Sgl) \\ &= 1 - (1 - 0.91) * (1 - 0.46) \\ &= 1 - (0.09 * 0.54) = 0.9514 \end{aligned}$$

Finally, the end-to-end system level security protection score for an attack is a composite of three scores = 0.19 * 0.46 * 0.9514 = 0.07which is only 7% or very low. This means that the entire camera system is prone to attacks. However, we can still use it due to our added security measures of a strengthened password, dual storage in the local and cloud based databases etc. Thus, one of the two paths needs to be secured to continue the required functionality: Path (a) Camera → Software → Local NAS, or Path (b) Camera → Software → Cloud Storage. Note that all past images will still be preserved even if the system is compromised up to the point of intrusion, e.g., if someone physically removes the microSD card on a security camera. If a home or business uses such a system, it may need multiple cameras so if one of them is compromised, others will continue the surveillance. An example is of 5 Pi cameras, with a shared local NAS and common cloud storages. The Security Score for the camera and software part is computed as 1 - (1 - 0.19*.46)^5 =0.36. The entire system security will be 0.36*0.9514 = 0.34 or 34%. This improves the total system security by almost 5X. Another way to achieve a better security is by making it harder to compromise a single camera system, e.g., by putting it in a cage so its microSD card can't be easily replaced. Then the probability of a physical attack goes from high to low, such that Sc = 1 - 0.3 * 0.9 = 1 - 0.27 = 0.73. The overall score for such a single camera system would be 0.73*0.46*0.9514 = 0.32, or 32%, which is almost same as our 5 parallel cameras system, albeit at a much cheaper cost. However, it also represents a single point of failure, so the real choice may be a combination of both. This can be achieved by using two secure camera systems in parallel, as redundancy is important to improve security.

7. SUMMARY

In this paper we review the scope of various IoT (Internet of Things) devices in the field that are bi-directionally connected to data-centers (in-house or cloud) via various networks. Then we look at the nature of security issues, and mechanisms to quantify risk associated with the complete hardware and software stack, with an example of a typical surveillance camera system. We calculated system security, and suggested ways to improve it. Our proposed method can be extended to evaluate any IoT system, and improve its end-to-end security profile.

8. ACKNOWLEDGEMENT

We wish to acknowledge help from Prof.Pramod Chandra P. Bhatt for giving the initial idea of a quantitative composite Trust Model for IoT devices. He has been a constant source of inspiration for this work, guiding us to completion. Prof. Bhatt can be reached at bhatt_pcp@consultant.com.

REFRENCES:

[1] N. K. Sehgal, S. Sohoni, Y. Xiong, D. Fritz, W. Mulia, and J. M. Acken, "A Cross Section of the Issues and Research Activities Related to Both Information Security and Cloud Computing," IETE Technical Review, Volume 28, Issue 4 [p. 279-291], 2011

[2] https://www.statista.com/statistics/370350/internet-of-things-installed-base-by-category/

[3] N. K. Sehgal, PCP. Bhatt, " Cloud Computing: Concepts and Practices," Springer Publications© 2018, ISBN 978-3-319-77839-6

[4] https://medium.com/@MarutiTech/what-is-serverless-architecture-what-are-its-criticisms-and-drawbacks-928659f9899a

[5] R. Sandhu, A.S. Sohal and S. K. Sood, "Identification of malicious Edge Devices in fog computing Environments," Information Security Journal: A Global Perspective, Volume 26, 2017, Issue 5.

[6] http://web4.uwindsor.ca/users/f/fbaki/85-222.nsf/0/b7d85091e772a10185256f84007be5c1/$FILE/Lecture_07_ch6_222_w05_s5.pdf

[7] https://www.raspberrypi.org/downloads/raspbian/

[8] https://pimylifeup.com/raspberry-pi-security-camera/

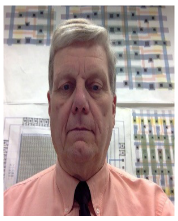
[9] https://www.sentinelone.com/blog/iot-botnet-attacks-mirai-source-code/

[10] https://aws.amazon.com/blogs/iot/understanding-the-aws-iot-security-model/

AUTHORS

NARESH is the Data-center Security and Privacy Director at Intel Corp. He has been with Intel for 28 years in various roles, including EDA development, Silicon Design Automation, Intel-HP Alliance management and for launching Virtualization technology on all Intel platforms. Naresh holds a PhD in Computer Engineering from Syracuse Univ., and MBA from Santa Clara Univ. He holds 5 patents and authored 30+ publications in the CAD domain.

SHIV SHANKAR is the founder of mapshalli.org, a social organization helping citizens and governments with technology. He was the Director of Enterprise Software Technology in Intel. He has a Master's degree in Mechanical Engineering from IIT, Madras, India.



JOHN M. ACKEN is a faculty member in the Electrical and Computer Engineering Department, Portland State University, Portland, OR. John received his BS and MS in Electrical Engineering from Oklahoma State University and Ph. D. in Electrical Engineering from Stanford University. He projects include technology and devices for information security and identity authentication. John has worked as an Electrical Engineer and Manager at several companies, including the US Army, Sandia National Labs in Albuquerque, New Mexico and Intel in Santa Clara, CA. John's time in the US Army was in the Army Security Agency, a branch of NSA during the Vietnam War.