GSMA

IoT SECURITY GUIDELINES

for IoT Service Ecosystems

Supported by

GSA

GLOBALPLATFORM®

IoT WORLD ALLIANCE®

simalliance

Security | Identity | Mobility

IoT Security Guidelines for IoT Service Ecosystem

Version 2.0

31 October 2017

This is a Non-binding Permanent Reference Document of the GSMA

Security Classification: Non-confidential

Access to and distribution of this document is restricted to the persons permitted by the security classification. This document is confidential to the Association and is subject to copyright protection. This document is to be used only for the purposes for which it has been supplied and information contained in it must not be disclosed or in any other way made available, in whole or in part, to persons other than those permitted under the security classification without the prior written approval of the Association.

Copyright Notice

Disclaimer

The GSM Association ("Association") makes no representation, warranty or undertaking (express or implied) with respect to and does not accept any responsibility for, and hereby disclaims liability for the accuracy or completeness or timeliness of the information contained in this document. The information contained in this document may be subject to change without prior notice.

Antitrust Notice

The information contain herein is in full compliance with the GSM Association's antitrust compliance policy.

Contents

1 Introduction

1.1 Introduction to the GSMA IoT Security Guideline Document Set

This document is one part of a set of GSMA security guideline documents that are intended to help the nascent “Internet of Things” (IoT) industry establish a common understanding of IoT security issues. The set of non-binding guideline documents promotes methodology for developing secure IoT Services to facilitate security best practices are implemented throughout the life cycle of the service. The documents provide recommendations on how to mitigate common security threats and weaknesses within IoT Services.

The structure of the GSMA security guideline document set is shown below. It is recommended that the overview document ‘CLP.11 IoT Security Guidelines Overview Document’ [1] is read as a primer before reading the supporting documents.

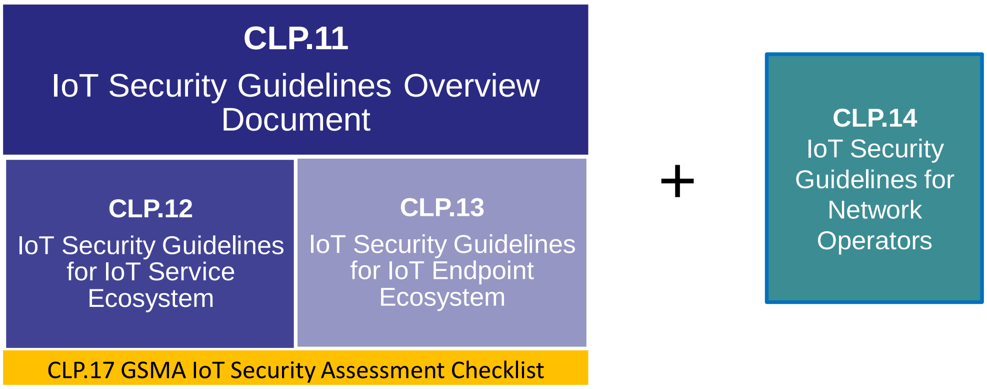
CLP.11
IoT Security Guidelines Overview Document
CLP.12
IoT Security Guidelines for IoT Service Ecosystem
CLP.13
IoT Security Guidelines for IoT Endpoint Ecosystem
CLP.17 GSMA IoT Security Assessment Checklist
+
CLP.14
IoT Security Guidelines for Network Operators

Figure 1 - GSMA IoT Security Guidelines Document Structure

Network Operators, IoT Service Providers and other partners in the IoT ecosystem are advised to read GSMA document CLP.14 “IoT Security Guidelines for Network Operators” [4] which provides top-level security guidelines for Network Operators who intend to provide services to IoT Service Providers to ensure system security and data privacy.

1.1.1 GSMA IoT Security Assessment Checklist

An assessment checklist is provided in document CLP.17 [13]. This document enables the suppliers of IoT products, services and components to self-assess the conformance of their products, services and components to the GSMA IoT Security Guidelines.

Completing a GSMA IoT Security Assessment Checklist [13] will allow an entity to demonstrate the security measures they have taken to protect their products, services and components from cybersecurity risks.

Assessment declarations can be made by submitting a completed declaration to the GSMA. Please see the following process on the GSMA website:

https://www.gsma.com/iot/future-iot-networks/iot-security-guidelines/

1.2 Document Purpose

This guide shall be used to evaluate all components in an IoT product or service from the Service Ecosystem perspective. The Service Ecosystem includes all components that make up the core of the IoT infrastructure. Components in this ecosystem are, for example, services, servers, database clusters, network elements, and other technologies used to drive the internal components of any product or service.

The scope of this document is limited to Recommendations pertaining to the design and implementation of IoT services and network elements.

This document is not intended to drive the creation new IoT specifications or standards, but will refer to currently available solutions, standards and best practice.

This document is not intended to accelerate the obsolescence of existing IoT Services. Backwards compatibility with the Network Operator's existing IoT Services should be maintained when they are considered to be adequately secured.

It is noted that adherence to national laws and regulations for a particular territory may, where necessary, overrule the guidelines stated in this document.

1.3 Intended Audience

The primary audience for this document are:

- IoT Service Providers - Enterprises or organisations who are looking to develop new and innovative connected products and services. Some of the many fields IoT Service Providers operate in include smart homes, smart cities, automotive, transport, heath, utilities and consumer electronics.
- IoT Endpoint Device Manufacturers - providers of IoT Endpoint Devices to IoT Service Providers to enable IoT Services.
- IoT Developers who build IoT Services on behalf of IoT Service Providers.
- Network Operators who provide services to IoT Service Providers.

1.4 Definitions

Term	Description
Access Control List	A list of permissions attached to a computing object
Access Point Name	Identifier of a network connection point to which an endpoint device attaches. They are associated with different service types, and in many cases are configured per network operator.
Attacker	A hacker, threat agent, threat actor, fraudster or other malicious threat to an IoT Service. This threat could come from an individual criminal, organised crime, terrorism, hostile governments and their agencies, industrial espionage, hacking groups, political activists, 'hobbyist' hackers, researchers, as well as unintentional security and privacy breaches.
Cloud	A network of remote servers on the internet that host, store, manage, and process applications and their data.
Container	A technology that makes it possible to run multiple isolated systems, or containers, on one host.

Term	Description
Embedded UICC (eUICC)	A UICC that supports remote provisioning of the network or service subscriptions it authenticates, as specified by GSMA.
End Customer	Means the consumer of the IoT Service provided by the IoT Service Provider. It is feasible that the End Customer and IoT Service Provider could be the same actor, for example a utility company.
Endpoint Ecosystem	Any configuration of low complexity devices, rich devices, and gateways that connect the physical world to the digital world in novel ways. See CLP.11 [1] for further information.
Forward Secrecy	A property of secure communication protocols: A secure communication protocol is said to have forward secrecy if compromise of long-term keys does not compromise past session keys.
Internet of Things	The Internet of Things describes the coordination of multiple machines, devices and appliances connected to the Internet through multiple networks. These devices include everyday objects such as tablets and consumer electronics, and other machines such as vehicles, monitors and sensors equipped with machine-to-machine (M2M) communications that allow them to send and receive data.
IoT Endpoint	A generic term for a complex IoT endpoint device or IoT gateway device.
IoT Service	Any computer program that leverages data from IoT devices to perform the service.
IoT Service Ecosystem	The set of services, platforms, protocols, and other technologies required to provide capabilities and collect data from Endpoints deployed in the field. See CLP.11 [1] for further information.
IoT Service Provider	Enterprises or organisations who are looking to develop new and innovative connected IoT products and services.
Network Operator	The operator and owner of the communication network that connects the IoT Endpoint Device to the IoT Service Ecosystem.
Organizational Root of Trust	A set of cryptographic policies and procedures that govern how identities, applications, and communications can and should be cryptographically secured.
Security Group	Acts as a virtual firewall that controls the traffic for one or more virtual server instance.
Trusted Computing Base	A Trusted Computing Base (TCB) is a conglomeration of algorithms, policies, and secrets within a product or service. The TCB acts as a module that allows the product or service to measure its own trustworthiness, gauge the authenticity of network peers, verify the integrity of messages sent and received to the product or service, and more. The TCB functions as the base security platform upon which secure products and services can be built. A TCB's components will change depending on the context (a hardware TCB for Endpoints, or a software TCB for cloud services), but the abstract goals, services, procedures, and policies should be very similar.
UICC	A Secure Element Platform specified in ETSI TS 102 221 that can support multiple standardized network or service authentication applications in cryptographically separated security domains. It may be embodied in embedded form factors specified in ETSI TS 102 671.
Virtual Private Network	Secured and logically separated partition of a network to allow dedicated usage by one particular customer set of service. So called because the VPN is Private from the rest of the network, and hence acts as a virtualised network in its own right

1.5 Abbreviations

Term	Description
3GPP	3rd Generation Project Partnership
ACL	Access Control List
API	Application Program Interface
APN	Access Point Name
CERTS	Computer Emergency Response Teams
CLP	GSMA's Connected Living Programme
DDoS	Distributed Denial of Service
GSMA	GSM Association
HSM	Hardware Security Module
IoT	Internet of Things
IP	Internet Protocol
SQL	Structured Query Language
TCB	Trusted Computing Base
VM	Virtual Machine
VPN	Virtual Private Network
WAF	Web Application Firewall

1.6 References

Ref	Doc Number	Title
[1]	CLP.11	IoT Security Guidelines Overview Document
[2]	CLP.12	IoT Security Guidelines for IoT Service Ecosystem
[3]	CLP.13	IoT Security Guidelines for IoT Endpoint Ecosystem
[4]	CLP.14	IoT Security Guidelines for Network Operators
[5]	n/a	OWASP Secure Application Design Project https://www.owasp.org
[6]	n/a	TCG Trusted Platform Module http://www.trustedcomputinggroup.org
[7]	n/a	TCG Guidance for Securing IoT http://www.trustedcomputinggroup.org
[8]	n/a	OAuth 2.0 http://oauth.net/2/
[9]		OpenID Foundation http://openid.net/foundation/
[10]	n/a	GSMA Mobile Connect https://mobileconnect.io/
[11]	GPC_SPE_034	GlobalPlatform Card Specification www.globalplatform.org/specificationscard.asp
[12]	GPD_SPE_010	GlobalPlatform TEE Internal Core API Specification

Ref	Doc Number	Title
		www.globalplatform.org/specificationsdevice.asp
[13]	CLP.17	GSMA IoT Security Assessment Checklist https://www.gsma.com/iot/iot-security-assessment/
[14]	n/a	ETSI TC SmartM2M specifications www.etsi.org
[15]	n/a	oneM2M Specifications www.onem2m.org
[16]	3GPP TS 33.220	Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA) www.3gpp.org

2 The Service Model

Modern IoT products and services require a Service Ecosystem to provide meaning, functionality, and value to Endpoints, Partners, and Users. Depending on the complexity of applications made available by the IoT offering, the infrastructure may be vast and composed of many disparate types of services and service access points. Alternatively, the infrastructure may be rudimentary for more straight-forward applications.

Regardless of the format, the Service Ecosystem acts as the nexus of functionality and communication for each core facet of the overall IoT technology. All other ecosystems are dependent on the Service Ecosystem for hierarchical authentication, connectivity to users, availability, management, and other tasks critical to the day-to-day operation of IoT. To accomplish these tasks, the Service Ecosystem is composed of any number of tiers required to fulfil the goals of the infrastructure. Database clusters, application servers, application proxy servers, and other types of infrastructure are example tiers that would be found in many given deployments. As implied in the diagram below, the Network and Endpoint Ecosystems are dependent on the core functionality of the Service Ecosystem.

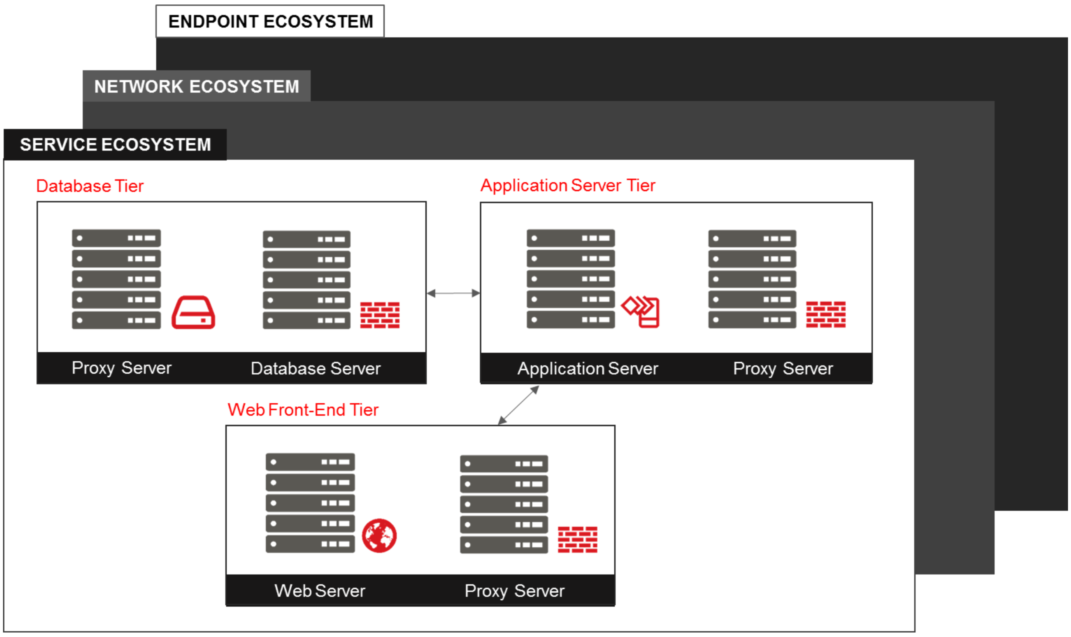

Figure 2 - Dependencies Hinged on the Service Ecosystem

Some examples of modern Service Ecosystems include, but are not limited to:

- Cloud Infrastructure based solutions
- Container based application deployments
- Traditional datacentre server environments
- Database clusters
- Web application framework service clusters

While each of these sample environments might seem widely variant in their design, topology, and implementation, they are based on the same theories regarding how information flows in and out of an application.

All modern computing systems require a point of entry, known as a service access point, into an application’s infrastructure. The internal subsystems that create content and context for that application must be able to process data from within secure, reliable environments and networks. Data must be stored somewhere, then returned to the service layer which responds or sends authorized commands down to various components within the same ecosystem, or other ecosystems and their associated networks.

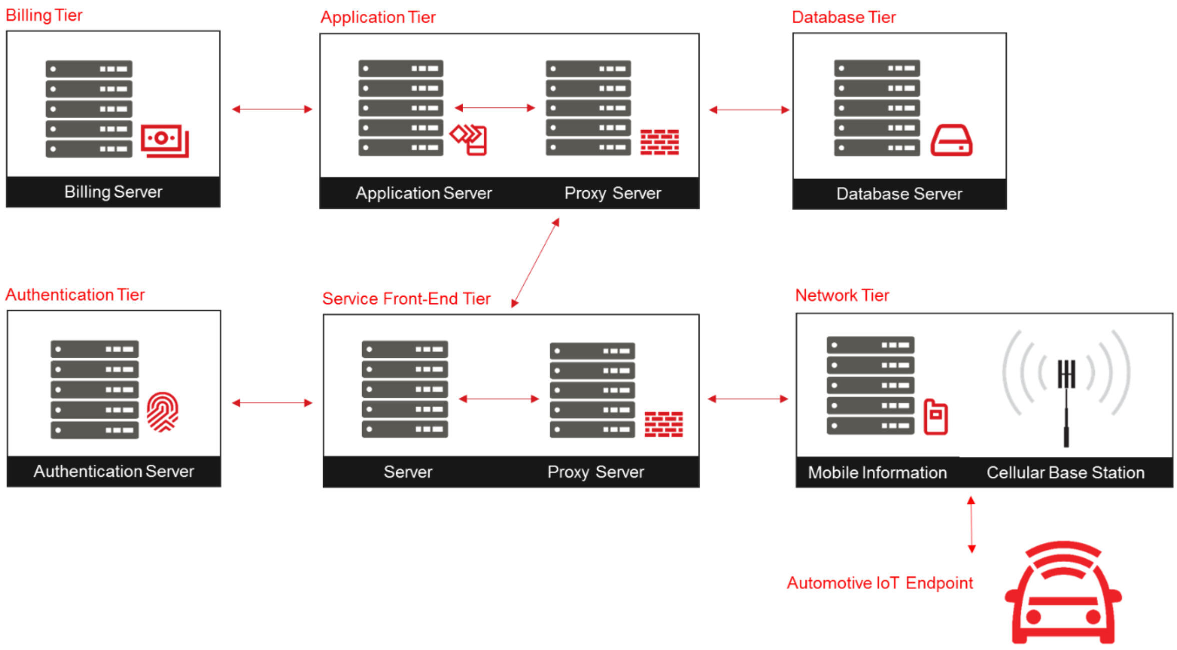

Figure 3 - Sample Service Ecosystem

Regardless of what technologies, modern or traditional, are used to implement this standard framework, information will be processed, served, and authenticated using proven protocols and technologies. While topologies and abstractions for processing environments have subtly changed to fit with modern requirements for speed, computational power, and storage, the technologies used to implement these innovations are, at their core, the same. For example, each tier typically contains a proxy or firewall system that manages connectivity to and from a set of servers of a specific type. Billing services will reside in a Billing Tier. Application Servers reside in a tier specific to the application. Database services must be managed within a Database Tier. These systems all work together based on the ingress and egress rules that are applied at the proxy servers.

As a result, the security model for the Service Ecosystem can be easily broken down into a set of components. These components will be discussed in this document.

3 The Security Model

Security in Service Endpoint environments can be designed using common pieces of infrastructure, strategies, and policies, regardless of the topology or innovations used to build an application architecture. Each aspect of the Service Ecosystem can be broken down into components. These components must be secured individually, but using similar methodologies.

For example, consider the common components in building a simple service that is capable of fielding queries and sending responses from and to endpoints, partners, and users. This model should contain, but not be limited to, the following tiers:

- A Web Service Tier
- An Application Server Tier
- A Database Tier
- An Authentication Tier
- A Network Tier
- Third party application tiers, such as a Billing tier

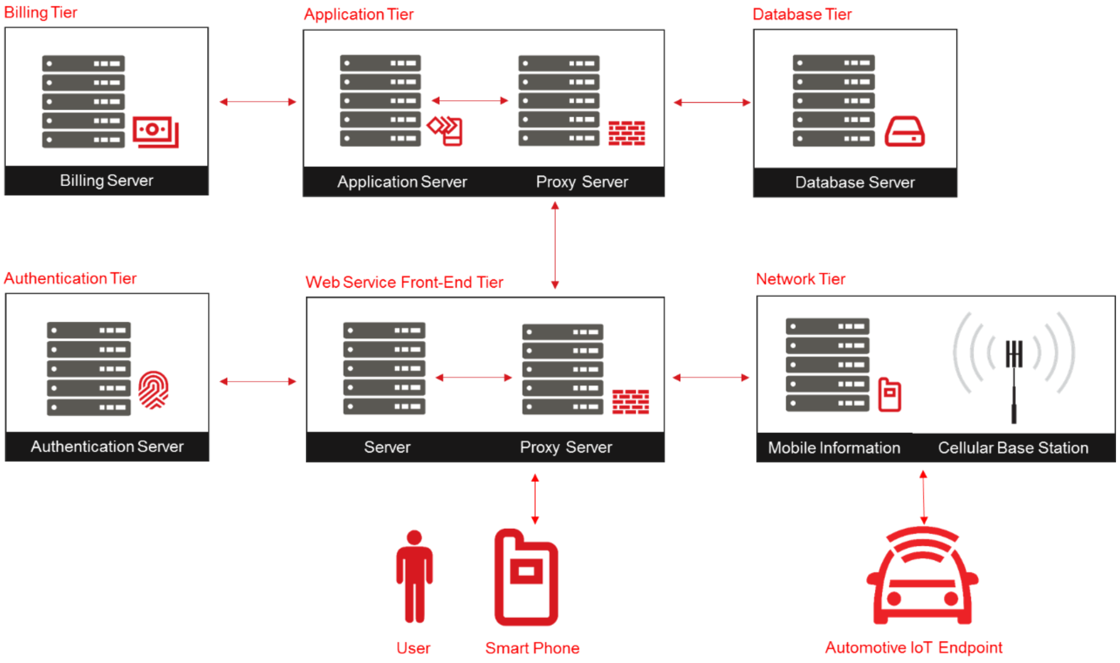

Figure 4 - An Example Service Ecosystem with Separate Tiers.

Even if there is only one server in each tier, it is more architecturally effective to separate each logical concept into its own tier. This also helps isolate one layer of technology from other layers in the event of a compromise, or if the system needs to scale up to serve more requests.

If a type of system is thought of from the perspective of a *type of tier*, it can be more easily secured, scaled on-demand, decommissioned and sunsetted. The only requirement is an

API that is versatile enough to be augmented or adjusted throughout the lifespan of the tier. Defining this API is out of the scope of this document. However, recommendations regarding high level security attributes of the API the organization either chooses or defines will be discussed here.

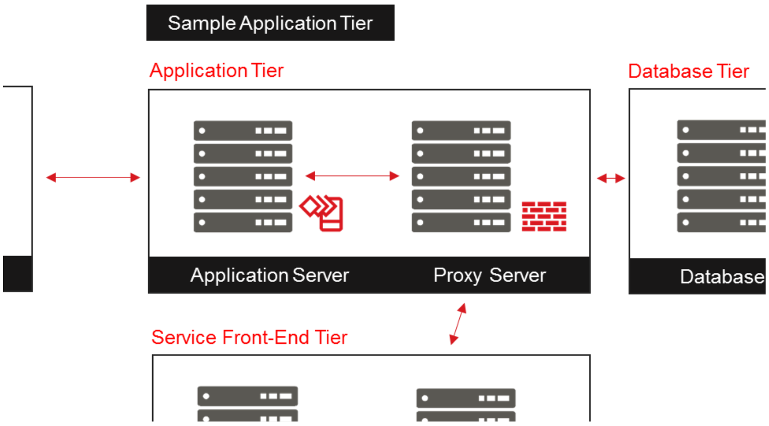

Figure 5 - An Application Tier Guarded by Firewall Technology

In the example above, a slightly more complete tier description is provided. The only augmentation that is needed to depict the tier is a proxy server. This proxy server is just a descriptor representing the actual security technology that will be employed within the tier. Regardless of whether the actual control is a hardware firewall, software firewall, Security Groups, Access Control Lists (ACL), or another technology, there will be a component that mandates ingress and egress controls on behalf of the tier.

When choosing or defining an API, the organization should consider existing specifications that may resolve the concerns of the engineering team. The organization should consider in particular the following specifications:

- ETSI SmartM2M TS 102 690, ETSI SmartM2M TS 102 921 [14]
- oneM2M TS-0001, oneM2M TS-0003 [15]
- 3GPP TS 33.220 [16]

For publicly accessible components, like the Service Front-End Tier, the only augmentation the model needs is an additional security component for:

- Distributed Denial of Service (DDoS) protection
- Load balancing
- Redundancy
- Optional Web Application Firewall (WAF) capability

The above technologies should be implemented for any service to function properly, and to ensure that the service they protect is made available even in the most resource constrained environments. Defining these components is out of the scope of this document, but can be further investigated by referring to the following entities:

- The Cloud Security Alliance
- NIST Cloud Computing Standards
- FedRAMP
- Cisco Network Management Guidelines

Other attributes required for the tier to function securely is the definition of the server, itself. This is defined by administrative, application, and operating system controls internal to the platform chosen by the engineering team.

While not exhaustive, a list of issues internal to the platform environment will be:

- Logging to a centralized log service
- Administrative Authentication and Authorization
- Communications security enforcement
- Data backup, restoration, and duplication
- Separation of application duties
- System Monitoring and Integrity

3.1 Networking Infrastructure Attacks

Adversaries attempting to compromise the Service Endpoint from the network perspective will presume that there are weaknesses in the way entities communicate, and vulnerabilities in services exposed through service access points. These attacks presume a privileged position on the network equates to a position of power over the communications channel.

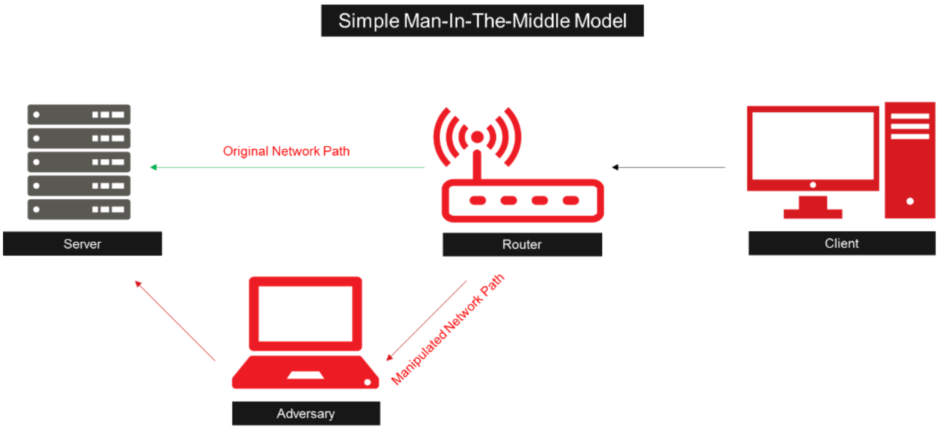

Figure 6 - An Example "Man In The Middle" Attack Model

The most common form of attack in this model is the Man-In-The-Middle (MITM) attack. This attack presumes that there is either no peer authentication, one-sided peer-authentication, or broken mutual authentication on the communications channel. An adversary's goal is to impersonate one side of the conversation to force the peer to perform actions on the adversary's behalf. This attack can be mitigated by enforcing mutual authentication, which requires a well-defined Organizational Root of Trust, a Trusted Computing Base (TCB), and a communications model.

Other attacks are, for example, attacks against Forward Secrecy, encryption communications analysis, and side-channel attacks. These must be mitigated using proper cryptography protocols, algorithms, and standards.

These attacks are difficult and require access to networking infrastructure either internally to an organization, in the core Internet infrastructure between an organization and its partners or Endpoint Ecosystem, or at the infrastructure near Endpoints. The simplest and most common attack is attempting to manipulate the network infrastructure of the Endpoint, such as the Wi-Fi, Ethernet, or cellular network, to gain a position of privilege between the Service and its peer.

Attacks against a single endpoint’s infrastructure are restricted to that endpoint, or the group of endpoints available in that physical location. Attacks against core internet infrastructure typically involve Border Gateway Protocol (BGP) hijacking, attacking a core router, or abusing the Domain Name Service (DNS) infrastructure. These attacks would provide a position of privilege more disassociated with a particular target, potentially allowing the attacker to have access to target many systems at once. Attacks against internal networking infrastructure require access to the internal network, which implies an insider attack or an existing position of privilege inside a corporation’s environment, which may imply an already deeper system compromise.

Regardless of which type of attack is utilized, this model is easy to mitigate using mutual authentication, forward secrecy, and appropriate cryptographic protocols and algorithms. Doing so will negate an attacker’s ability to abuse this infrastructure, or will skyrocket the cost of this type of attack to a price point that is infeasible for the common attacker to implement.

3.2 Cloud or Container Infrastructure Attacks

These attacks presume a position of privilege on the Cloud or Container infrastructure environment. For example, if an adversary is able to compromise a Cloud service network, they may have access to hosts running guest Virtual Machine (VM) systems. This would allow the adversary to inspect and modify running VM systems. The adversary may have specific targets in mind, or may have gotten lucky and compromised a Cloud service provider just for the access to many different types of systems with valuable data.

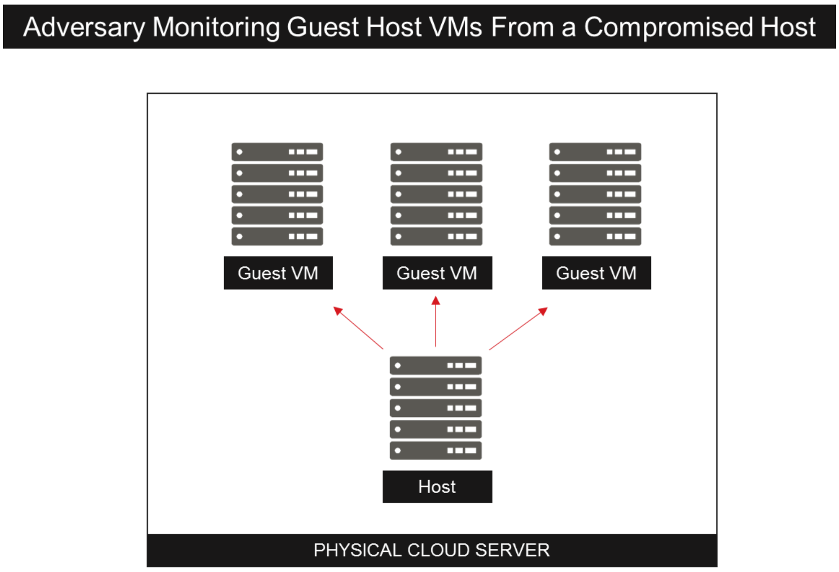

Figure 7 - Example VM Attack Model

Another Cloud or Container infrastructure attack presumes that the adversary has control over a VM on the same physical server as the target VM. The adversary may then use several methodologies to compromise other VMs on a physical server. They could:

- Use a vulnerability in the VM infrastructure to break out of a Guest into a Host system
- Use a side-channel attack to infer secret keys from another Guest VM
- Consume excessive resources on the physical server to force a target VM to migrate to a physical server that the attacker has more control over

Regardless of the attack model utilized, there is little that a business can do to guard against this risk. Instead, the Cloud service provider must implement adequate functionality to reduce the probability that an attacker can subvert the Cloud or Container infrastructure.

One way to reduce this risk is to implement a Container based architecture that limits each container to one specific user and a unique cryptographic identity. While this is a highly resource intensive activity, and may incur additional cost, it will mitigate the ability for an adversary to abuse the VM infrastructure, to gain access to multiple users or multiple services at once.

While a position of privilege in a Cloud or Container environment is a critical threat to applications executing in guest VMs, it takes a high degree of skill, time, and resources to gain access to this position. Once access is acquired, the adversary must maintain it long enough to identify which system contains the VM that is relevant to their interests. Furthermore, they must be able to monitor or alter that VM without being detected by the Cloud services provider’s incident subsystem. This may pose a significant challenge, and should diminish the likelihood of a compromise.

However, it is notable that this type of compromise is largely undetectable by the guest VM, or an application running on top of it. Thus, metrics may be gathered that reveal anomalies in the behaviour of a particular Cloud VM or Container, but it may be extremely difficult to identify whether or not a compromise actually occurred. This is because any adversary with sufficient privilege at the host layer of the VM infrastructure would be able to manipulate the guest to make it difficult for it to detect manipulation.

Attacks from guest to guest are exceptionally difficult to detect, even by the Cloud service provider. It is important to note, however, that these attacks are largely theoretical in nature. While side channel attacks are possible, whether they are practical is up for debate as these attacks require a level of consistency in the underlying execution platform that is not guaranteed in a real world environment. Furthermore, escalation attacks from guests to hosts in a VM, container, or hypervisor environment are difficult to find and even more difficult to exploit. This makes it much less likely that a vulnerability will result in exploitation of a mass amount of guests, or a specific target.

Therefore, while this is a significant position of privilege for attackers, the likelihood of a successful attack is low as the difficulty, cost, and opportunity make exploitation mostly impractical.

3.3 Application Service Attacks

While application execution architecture discussions are largely out of scope with respect to this document, it is important to note that this layer represents the greatest risk of an attack. If the Service Ecosystem has been configured correctly, as is recommended in this guide, attackers will migrate away from network infrastructure attacks to the application, itself.

The application presents the largest layer of complexity in any product or service, and always contains the potential for an adversary to increase their privilege through multiple tiers of technology. Therefore, while the goal of this document is to drive the focus of an attack away from the network infrastructure, it is driving the focus largely *to the one place* where success is far more likely.

To diminish the potential for attack, please review many of the well documented pieces of literature on application security (for example the OWASP Secure Application Design Project [5]), to implement the application execution architecture as securely as possible.

3.4 Privacy

While partner systems are designed to consume data/metrics or other user-centric components to provide added value to the overall system, there is never a guarantee as to the level of security implemented by the partner. Rather than simply pass information to a third party, it is necessary to evaluate what types of data should be handed over, what the tangible return should be, and how that information shall be protected.

Legal liability can be diminished through contracts and insurance clauses, however, losing customers can occur due to a third party’s failure. Rather than risking such a loss of business, an organization should evaluate third party engineering teams to determine what level of security they apply to their infrastructure, applications, and APIs. If the level of security is not sufficient, it is recommended to look for alternative partners.

3.5 Malicious Objects

Third party systems are designed to present information or multimedia to consumers. One obvious way to accomplish this is through advertising. Various types of files are complex in their structure, and are difficult for software to parse correctly. Advertising networks are an interesting channel for the distribution of malware. Content Distribution Networks (CDNs) also represent potential channels for malware distribution. Any system that offers complex multimedia types, or bundles of code (either web or executable) for the purposes of rendering dynamic information, can traffic malware.

Therefore, it is imperative that the business evaluate the different types of technological offerings that will be passed through a particular channel. The business must decide what to allow and what is too excessive to hand off to their customers. For example, an advertising firm may want to traffic Java code to client systems over a proxy service application offered to partners by the IoT business. The business will need to decide whether the client systems running on certain environments are more susceptible to attack from Java technology. If this is found to be true, the business may want to disallow Java, but allow other technology, such as Hypertext Mark-up Language (HTML) to pass through.

Since malware comes in many forms, ranging from polymorphous file types, to Adobe Flash, Java, and multimedia exploits, there is no single uniform way to guarantee the end-user’s safety. A simple solution would be for the engineering team to enforce a policy on what technologies *are* used over their channels, and *how* their users will be impacted. Monitoring subsystems can be put into place, as well as sandboxes, to ensure that any object rendered on a client system is less subject to abuse.

3.6 Authentication and Authorization

Partners often offer services that are only specific to a subset of users. This may include paid services that a user can optionally subscribe to. This also may represent a way that a user can authenticate to the system, but by using credentials shared with a separate, well known, technology, such as existing authentication APIs from network service providers, social network infrastructure, and existing M2M or IoT management entities.

While these are excellent ways to share technology across platforms, engineers must ensure that technology is not inadvertently consuming credentials that can be used to abuse permissions not expressly granted to a third party service. For example, certain platform APIs allow the constraining of permissions to a class that is either accepted or denied by the user. This allows the user to tune the experience to one that is suitable for their specific privacy needs. If the platform is unable to offer granular security permissions, then it should list what technologies it *does* want access to.

It is necessary for the engineering team to request of their partners that the offering enable granular permissions to ensure that revocation of a service does not inadvertently allow a window of exposure of that users data that continues on even after the subscription is revoked.

3.7 False Positives and False Negatives

While monitoring and logging services are exceptional ways to augment an existing security infrastructure, they must be carefully evaluated for false positives and false negatives. Because these systems only interpret data originating from various ecosystems within an IoT product or service, and these systems are not developed by the in-house engineering team, they can only offer artificial insight into an event. They may not however be able to accurately distinguish whether an adversarial event is actually occurring.

As a result, it is important to gauge the IT and engineering teams to determine if a suspect event is, in fact, attributable to malicious behaviour. This will help to negate the potential for the monitoring team to disallow a legitimate user's access to the system. If this process is automated, and the process is incorrect, many users may be shut out of their legitimate service due to a false positive that can be attributed to an anomaly in the client application or infrastructure. When a critical event is occurring that is questionable, the IT and engineering teams should take a look at the data to evaluate whether there is indeed an attack.

Additionally, engineers must be careful to model information acquired through analogue channels. False positives and false negatives, especially in ecosystems where data must be processed at exceptionally high rates, can have significant consequences if the application doesn't properly evaluate the safest course of action in the event that the acquired data cannot be entirely trusted. It is notable that with sufficient time, technology, and expertise, all analogue data can be impersonated to a digital system.

4 Frequently Asked Security Questions

Service security is broken down into recommendations by priority in this document. But, for practical use, it is more beneficial to evaluate recommendations from a practical starting point. Engineers typically start building a list of recommendations based on a technological or business-influenced goal. This section outlines common goals from an endpoint perspective, and which recommendations are relevant toward achieving those goals.

4.1 How do we Combat Cloning?

Differentiating between valid devices manufactured by the IoT Service Provider and devices that are reproductions or "rip offs" (clones) is a challenging feat. No IoT Service Provider wants to provide services for unauthorized endpoints, as Service Providers have to pay for the CPU time, bandwidth, disk storage, and other resources. The organization must pay regardless of whether the device was manufactured by the IoT Service Provider or not.

Furthermore, the organization must be able to discern whether their endpoint architecture is being subverted. This enables the organization to react to a device that has been *cloned* into multiple instances of the same device. This could be done by an unscrupulous manufacturer, or an adversary trying to impersonate a particular user.

Review the following recommendations for assistance on using the Service to combat cloning:

- Define an Organizational Root of Trust
- Use Network Authentication Services
- Force Authentication Through the Service Ecosystem
- Define Application Layer Authentication and Authorization

4.2 How Are Users Authenticated via the Endpoint?

One of the most important concepts in IoT is the separation of endpoint authentication from user authentication. An endpoint can be authenticated by its Trusted Computing Base, but how the *user* is authenticated is a separate process that relies on the endpoint's TCB for communications security. What is most important about this abstraction is evaluating how trustworthy the communications channel is for user authentication.

For example, if the trustworthiness of an endpoint is low because there is no endpoint TCB, or a weak endpoint TCB implementation is used, the user authentication mechanism that relies on endpoint software/firmware to run cannot be trusted. This means that any user authenticating through an endpoint device cannot be considered authenticated.

From a different perspective, a well architected endpoint TCB can poorly authenticate the end-user if the authentication scheme is easily bypassable. Thus, the service ecosystem must rely on the endpoint's trustworthiness as well as the authentication mechanism's implementation to ensure the service ecosystem can make guarantees about whether the correct user is logged into the system.

Consider the following recommendations for assistance in dealing with these complexities:

- Implement a Service Trusted Computing Base
- Define an Organizational Root of Trust
- Define a Clear Authorization Model
- Use Network Authentication Services
- Force Authentication Through the Service Ecosystem
- Enforce Strong Password Policy
- Define Application Layer Authentication and Authorization

4.3 How can the Service Identify Anomalous Endpoint Behaviour?

One of the most challenging aspects of managing endpoints in a distributed IoT network is determining whether or not an endpoint is behaving in an abnormal fashion. This is not only important from a security perspective, but from a reliability perspective. Often, anomalous behaviour can be indicative of a problem with the firmware or hardware, and may be a sign that the organization should prepare to remediate an unexpected problem. However, if the behaviour is isolated into a part of the network that cannot be analysed by the IoT Service Provider, these metrics will be lost, leaving the organization at far less of an advantage.

Resolving this issue requires the ability to inspect behaviour on the endpoint, the network layer, and the service ecosystem. However, if the right infrastructure, services, and partnerships are not built to gather these data points, the organization won't have the information required to make a determination as to whether there is a problem, or whether a problem is related to security or reliability.

Evaluate the following recommendations from the service ecosystem perspective:

- Define a Security Infrastructure for Systems Exposed to the Public Internet
- Define a Systems Logging and Monitoring Approach
- Define a Communications Model
- Use Network Authentication Services
- Implement Input Validation
- Implement Output Filtering
- Use Partner-Enhanced Monitoring Services

- Use a Private APN for Wireless Connectivity
- Define a False Negative/Positive Assessment Model

4.4 How can the Service Restrict an Abnormally Behaving Endpoint?

Once an endpoint is identified as behaving abnormally, the service should make decisions as to what resources should be limited or restricted. This question is relevant to every layer of the service infrastructure.

For example, a cellular-enabled endpoint that constantly connects to and disconnects from the mobile network in a frantic loop should be forcibly disabled until the erratic behaviour is resolved. Another useful example is a compromised endpoint that an adversary is using to try and attack back-end services. In this scenario, the back-end services should disallow the abusive endpoint from reaching the services at all.

How to handle each scenario is up to the IoT Service Provider, and depends on their business goals, and how incidents should be handled. To assist with developing these guidelines, consider the following recommendations:

- Define an Organizational Root of Trust
- Define a Security Infrastructure for Systems Exposed to the Public Internet
- Define an Incident Response Model
- Define a Recovery Model
- Define a Sunsetting Model
- Define a Communications Model
- Define a Breach Policy for Exposed Data
- Force Authentication Through the Service Ecosystem
- Use a Private APN for Wireless Connectivity
- Define a False Negative/Positive Assessment Model

4.5 How Can I Determine Whether a Server or Service Has Been Hacked?

While endpoint anomalies are more esoteric and require a vast amount of behavioural analytics to catch a majority of attacks, the service ecosystem is more straight forward. Services and servers are deployed within an environment that is tightly controlled by the IoT Service Provider or their partners that manage the cloud or server infrastructure. Thus, the organization and its partners can use readily available monitoring and diagnostic systems to identify and contain potential problems.

Review the following recommendations for assistance:

- Define an Administration Model
- Define a Systems Logging and Monitoring Approach
- Define an Incident Response Model
- Implement Input Validation
- Implement Output Filtering

4.6 What Can I Do Once A Server Has Been Hacked?

When a server has been identified as compromised, the administration team needs to resolve the issue as quickly and efficiently as possible. The complexity in doing so often arises from determining what resources, information, and accounts have been put at risk. In some poorly architected environments, the effects of a compromise are not often quantifiable. Therefore, the organization *must* implement a plan to resolve the security vulnerability *and* secure at-risk assets in the field, in parallel. Once the ecosystem and vulnerability have been secured, the organization can proceed with a plan to rebuild the affected technology.

Review the following recommendations for more info:

- Define an Incident Response Model
- Define a Recovery Model
- Define a Sunsetting Model
- Define a Set of Security Classifications
- Define Classifications for Sets of Data Types

4.7 How Should Administrators Interact With Servers and Services?

Development an administrative model that doesn't put the service ecosystem at risk is an important part of the architecture of an IoT service. There are multiple layers of administration, and each layer should be considered by the engineering and security teams. For example, administrators that govern the server (regardless of whether a virtual, micro-service, or uni-kernel architecture is used) must be able to interact with live servers through a reliable and secure communications channel. Administrators that govern the web application often interact with the application over the same web communications layer, but through a speciality application embedded in the code.

Regardless of the administrative need, the interface should be restricted-access to limit the ability for adversaries to interact with or abuse the technology. Consider the following resources:

- Define a Security Infrastructure for Systems Exposed to the Public Internet
- Define an Administration Model
- Define a Clear Authorization Model
- Define a Communications Model
- Use a Private APN for Wireless Connectivity

4.8 How Can the Service Architecture Limit the Impact of a Compromise?

One fascinating attribute of an IoT network is its unique ability to attach services to specific consumers. In web services, each user must be given the ability to interact with the service from any type of device or, potentially, from anywhere in the world. This is not true of IoT technology. IoT technology typically requires a specific endpoint device to interact with IoT services. Because of this difference, server ecosystem architects can leverage the one-to-one relationship between endpoints and consumers to restrict an endpoint's access to back-end data.

Consider the scenario where an endpoint is pushing sensor metrics to a back-end service. In a micro-service architecture, the service ecosystem may deploy a specific micro-service or uni-kernel to handle a particular consumer. Using this architecture, the engineer can ensure that the micro-service is provisioned *only* with the resources and access capabilities required to deliver data and services *specific to the individual consumer*.

This means that if a service is compromised, and the endpoint is the only technology that can communicate with that specific service, there is zero additional benefit to compromising that service as the access gained from the compromise will be limited to the resources that would *already* be available to the endpoint. In essence, there is a zero sum gain from the attack.

Review the following recommendations for assistance:

- Implement a Service Trusted Computing Base
- Define a Bootstrap Method
- Define a Security Infrastructure for Systems Exposed to the Public Internet
- Define a Persistent Storage Model
- Define an Administration Model
- Define a Sunsetting Model
- Define a Clear Authorization Model
- Provision Servers Where Possible
- Define an Application Execution Environment
- Virtual Machine Compromises

4.9 How Can The Service Architecture Reduce Data Loss During a Compromise?

Another interesting attribute of IoT architecture is the reduction of data loss. This is similar to how services can be isolated to a specific user. Data can also be isolated to a specific user once the user has been authenticated. However, data storage cannot easily be implemented on a per-user basis because of the expense of database and storage infrastructure.

Instead, unique tokens should be provisioned to services that then act on behalf of a specific user within the storage infrastructure. This way, an attacker with access to the data storage environment may be able to connect to the service, but should not be able to interact with, retrieve, or alter user data other than the user that has been compromised.

From the perspective of the network layer, reducing the flow of traffic from the server ecosystem out to the Internet is also a requirement. Egress controls force an adversary to traffic intellectual property or customer data through specific channels. This can either increase the difficulty of moving large amounts of data, or force it through communications layers that can detect and cut off communication during incidents.

For more information, consider the following recommendations:

- Define a Bootstrap Method
- Define a Security Infrastructure for Systems Exposed to the Public Internet
- Define a Persistent Storage Model

- Define a Set of Security Classifications
- Define Classifications for Sets of Data Types
- Provision Servers Where Possible
- Define an Application Execution Environment
- Default-Open or Fail-Open Firewall Rules

4.10 How Can the Service Architecture Limit Connectivity from Unauthorized Users?

A benefit of leveraging common IoT architectures is restricting the ability for unauthorized Internet users to directly connect to back-end services. Most web applications do not have this luxury, and must be available for public use. In IoT, however, because the endpoint is the entity that must connect to a particular service, Virtual Private Network (VPN) can be used to restrict who has access to back-end services. This can be implemented over standard Internet protocols, or can be implemented using mobile services, such as the Private APN. Review the following recommendations for more information:

- Define a Security Infrastructure for Systems Exposed to the Public Internet
- Use a Private APN for Wireless Connectivity

4.11 How to Reduce the Likelihood of Remote Exploitation?

Remote exploitation of web applications and services is a constant concern from infrastructure administrators. Ensuring that adversaries don’t have a route into the internal network, or simply to valuable resources, is a daily battle. The only way to reduce the potential for adversaries to compromise the service ecosystem is to decrease the potential targets into a manageable set of services that can be quickly and easily maintained. The second most important augmentation to the architecture is the design of the underlying architecture: the execution architecture, operating system configuration, deployment toolchain, programming language security, and other options that define how securely an application may run. These options can be the difference between an application crash and an infrastructure compromise.

For more information on reducing the potential for remote exploitation, please see:

- Define a Security Infrastructure for Systems Exposed to the Public Internet
- Define an Update Model
- Implement Input Validation
- Implement Output Filtering
- Default-Open or Fail-Open Firewall Rules
- Define an Application Execution Environment
- Rowhammer and Similar Attacks
- Virtual Machine Compromises

4.12 How Can the Service Manage User Privacy?

As IoT Service Providers grow, they will invariably build partnerships with organizations that will utilize consumer data in innovative ways. However, this data comes at a cost to the consumer's privacy. Consumers should have a right to determine what data is shared with partners and how it will be used. Also, partners should be required to use the data in specific ways. Authorization models can assist with this, but this implies a far larger discussion around privacy, legal repercussions, business insurance, and more.

To start the discussion within your organization, please review the following recommendations:

- Define a Set of Security Classifications
- Define Classifications for Sets of Data Types
- Define a Clear Authorization Model
- Define a Breach Policy for Exposed Data
- Evaluate the Communications Privacy Model
- Define a Third-Party Data Distribution Policy
- Build a Third-Party Data Filter
- Build an API for Users to Control Privacy Attributes

4.13 How Can a Service Improve Its Availability?

Denial of Service (DoS) attacks or Distributed Denial of Service (DDoS) attacks are so commonplace in the modern Internet that every company should be prepared to face a major attack of this class, and should be able to stay online even under prolonged attacks. The reason why these attacks have become so commonplace is that they lack very little skill to execute and the tools to implement such an attack are readily available online. In fact, there are service online where a malicious party can pay an attacker to implement a DDoS attack against a particular target.

As a result, entirely new models for the availability of services have been built to combat this threat. Consider the following recommendations when building out the service ecosystem:

- Define a Security Infrastructure for Systems Exposed to the Public Internet
- Define a Systems Logging and Monitoring Approach
- Define an Incident Response Model
- Define a Recovery Model
- Define a Communications Model
- Default-Open or Fail-Open Firewall Rules

5 Critical Recommendations

When developing a secure endpoint, the following recommendations should always be implemented. The following critical recommendations define a secure endpoint architecture. Without these recommendations, the endpoint will have an incomplete security profile that will be abused by an adversary.

5.1 Implement a Service Trusted Computing Base

A Trusted Computing Base (TCB) is a set of hardware, software, protocols, and policies. A TCB must be the basis of any given computing platform, and must define the environment in which an application can run reliably, securely, and with high quality.

A TCB can be built and deployed for any given class of system, such as Mobile Equipment (smartphones), IoT Endpoints, and even servers living in a Service Ecosystem. TCBs are all composed of similar technologies. Yet, depending on the class of system, those technologies may take on very different characteristics. For example, bootstrapping a TCB in a cloud server will look vastly different than bootstrapping an endpoint.

Building a TCB in a Service Ecosystem means defining the way an application image shall be rolled out. An image in this context represents the raw binary data that comprises an application executable, its configuration files, and its metadata. These things together are commonly referred to as an application image, or simply image. In most modern Service Ecosystems, systems will be replicated, powered up, or spun down on demand to reactively scale with changes in the computing environment. This means that a TCB must define a way to allow systems to scale effectively while maintaining a persistent security model.

To do this correctly, the team must:

- Standardize the computing platform:
  - Choose a set of physical server models
  - Select a set of Cloud platforms or Virtual Machine (VM) images
- Define the set of applications, libraries, and configuration files to be ran on the computing platform:
  - Define a container environment, if applicable
- Generate an application image, composed of the set defined above
- Cryptographically sign an archive of the image using the Tier TCB Signing Key
- Securely store the archive and the signature

Performing this set of tasks will result in an approved application image that can be deployed in a specific tier. Each tier will have a different hardware and application model that works best for that specific tier. For example, database hardware has vastly different performance and storage needs than an application tier. A storage tier will have similar hardware storage requirements as a database tier, but will have different performance requirements. After standardizing each tier’s definition, the result is an image that can be deployed and verified on each hardware platform.

The difficulty in deploying a TCB comes from:

- Setting up an Organizational Root of Trust to manage cryptographic signing of images
- Setting up a procedure for signing each image
- Setting up a procedure for verifying each image
- Setting up a procedure for rolling out images in an automated way, but with image verification

Please consider using material from the following organizations to assist with this recommendation:

- GlobalPlatform Card Specification [11]
- Trusted Computing Group's TPM Specification [6]
- GlobalPlatform TEE Internal Core API Specification [12]

5.1.1 Risk

Without a well-defined Trusted Computing Base, computing platforms are unable to verify that it is currently running in a configuration approved by the engineering team. This is important as the application subsystem must be able to determine if it has been compromised by an adversary. A TCB can be used to remediate this risk, as well as provide a security layer for all network communications.

5.2 Define an Organizational Root of Trust

An Organizational Root of Trust is a certificate or public-key based system for authenticating computing platform entities in an organization. Each computing platform in a Service Ecosystem must be cryptographically authenticated during network communications. This diminishes the ability for an insider, or someone within a privileged network position, to impersonate or otherwise abuse the trust of a privileged system.

To build an Organizational Root of Trust, simply perform the following actions:

- Build or acquire, for example, a Hardware Security Module (HSM) to store the organizational root secret
- Generate a root secret and/or certificate
- Ensure the private facet of the secret is stored securely
- Generate a set of one or more signing keys to be used for Tier TCB signing key
- Sign the public facet of the signing key with the organizational root
- Ensure these keys cannot be used without authentication and authorization from the business and engineering leads

Every time a new Tier system is defined, its unique cryptographic key or certificate can now be signed by the signing key. If another system connects to this new system, it can validate the identity of the system by verifying the chain of trust defined by the organizational root.

It will cryptographically validate that the messages were signed by the public key representing the system. Then, it will verify the signature the signing key generated of that system's unique public key. Then, the client should verify that the signing key was indeed the signing key authenticated by the organizational root.

Because each set of certificates or secrets is restricted to fewer and fewer individuals in the organization, and the policies and procedures defined should restrict who can use those secrets and when, each level of trust should increase as the client descends through the root chain.

A service must be defined that presents authentication capabilities to authorized peers within the Service Ecosystem. For example, authentication using the certificate or secret chain can’t be used on its own to guarantee security. A service must be made available that verifies whether certificates have been revoked, or are currently valid. Another service may need to be used to authenticate the identities of servers or services with a short lifespan, depending on the requirements of the underlying infrastructure.

During the definition of the Root of Trust, consider that:

- Each secret must be guarded from abuse
- Internal use of each secret must be verifiably tracked and monitored
- Each individual approved to utilize a secret must use multi-factor authentication when accessing the secret(s)
- Defining a set of policies and procedures that enforce consistent and secure usage can be challenging
- Building a process to sunset or revoke a certificate can be challenging
- Identifying whether a key has been abused can be challenging
- Choosing the correct set of cryptographic algorithms may be non-intuitive

For further reading on the Root of Trust concept, please consider the following sources of information:

- Trusted Computing Group
    - TPM Specification [6]
    - TCG Guidance for Securing IoT [7]
    - ISO 11889
- PKI Specifications
    - RFC 2510
    - RFC 3647

5.2.1 Risk

The risk of not using an organizational root of trust is that any compromise to a single key can result in compromise of the entire ecosystem. By separating the organization into a hierarchy, and deploying separate keys for the hierarchy, keys can be cycled at regular intervals and according to the priority of the application or sub-organization the key relates to.

5.3 Define a Bootstrap Method

In order for an application to run properly, it must be loaded and executed in a consistent way on a reliable, high quality, and secure platform. The TCB defines how to formulate this platform, but the Bootstrap model defines how the application shall be ran on top of it.

To define a Bootstrap model effectively, the following must be considered:

- Define an API that allows the application to cryptographically identify itself to its peers
  - Consider utilizing an existing API defined by a trusted industry leader
- Define how the application will authenticate Endpoints, Service Peers, and Partners
- Define what the application's configuration should look like
- Enforce each different application to have a unique identity, especially applications running on separate tiers

While it may seem intuitive that an application should cryptographically identify itself to its peers, and it may not need an API to do so, the process in production is slightly non-intuitive. This is because in the bootstrap model, one must consider *how* the cryptographic identity is provisioned *to* the application. How does the application acquire its identity? Is the identity acquired securely? What is the process for revoking secrets that the identity will use in the event the secrets must be updated or altered?

At run-time, applications require certain resources to execute effectively. The application must be able to communicate and perform mutual authentication with all external services, endpoints, and partners that are involved in this process.

An application's configuration often determines how secure it is in production. A configuration should be enforced, and presented read-only to an application. The application, or someone abusing the application infrastructure, should not be able to simply alter the configuration of an application.

Use the Organizational Root of Trust to define trust models for each Tier deployed in the overall ecosystem. This will allow each separate application to have a unique cryptographic identity. This will provide peers with the ability to differentiate between a database service and an application service, for instance.

5.3.1 Risk

Without a well-defined Bootstrap Model, the system will have no way of verifying each layer it requires to operate. In essence, there is no layering of trust on each facet of the overall technology. This lack of trust layers introduces complexity that can result in gaps that may be abused by adversaries.

5.4 Define a Security Infrastructure for Systems Exposed to the Public Internet

For publicly accessible services, several pieces of security and reliability technology are required to maintain the availability, confidentiality, and integrity of the service:

- DDoS-resistant infrastructure
- Load-Balancing infrastructure
- Redundancy systems
- Web Application Firewalls (optional)
- Traditional Firewalls

These additional technologies should be placed in front of the application's tier to ensure that it can't be manipulated by public attackers. While the communication security model will remediate or mitigate the potential for an anonymous third party to access the system, these technologies will diminish the ability for the adversary to make the system unavailable.

Front-end security should be applied to *all* protocols implemented by the services. For example, if the service is available over IPv4 and IPv6, the same security constraints should be applied to the service over both protocols. If a service is accessible over TCP as well as Stream Control Transmission Protocol (SCTP), the security constraints should be applied to both of those protocols as well. Ports that do not offer public services pinned to the IoT product or service should not be accessible.

Ensure that both ingress *and* egress filtering are managed, whenever possible. While ingress filtering stops a range of attacks, any attack against a publicly accessible service can still result in a compromise of the service ecosystem. Egress filtering is imperative, at this point, to ensure that a compromised component of the service ecosystem cannot be used by an attacker to move laterally within the ecosystem. In addition, egress filtering helps to complicate the ability for attackers to exfiltrate critical data from the ecosystem to adversary-controlled servers, leaving more time for administrators to identify and isolate the attacker.

Several organizations offer these services in a simple API model that can be dropped into a given technology. This allows the technology to be used with little effort. Not much engineering effort is required beyond signing up and configuring the application within the service provider’s system. Consult with your service provider to determine the best way to implement their security technology for your environment.

Please consider using material from the following organizations to assist with this recommendation:

- Amazon Best Practices for DDoS Resiliency:
  - https://d0.awsstatic.com/whitepapers/DDoS_White_Paper_June2015.pdf
- Arbor Networks DDoS Mitigation Best Practices:
  - https://www.arbornetworks.com/images/documents/Arbor%20Insights/AI_DDoSMitigation_EN2013.pdf
- Cisco DDoS Defence Guide:
  - http://www.cisco.com/web/about/security/intelligence/guide_ddos_defense.html

5.4.1 Risk

Secure infrastructure for public-facing services and applications is imperative due to the volatile nature of the Internet. Random DDoS attacks occur often, and for little reason. DDoS services can be purchased in the “underground market” for several hundred dollars. Thus, adversaries of the business, or a business’s customer, will not be the only perpetrator of such attacks. Random attacks can occur just to see if it is possible to take a system down. It is better to be prepared against such attacks to ensure that critical IoT services are not unexpectedly brought down. Availability is a critical component of an IoT product or service.

5.5 Define a Persistent Storage Model

Application environments in modern computing are often ephemeral, such as Container based systems, or Cloud environments. As a result, the storage allocated to these systems is not large enough, nor designed to be made available long term, for the application to use these technologies as persistent storage. In addition, these systems may be defined as on-demand entities, and may have no semblance of centralization. In other words, there is no way for the other systems to define *which system* has enough storage for persistent use.

This is why central storage systems are imperative, and why they must be carefully secured. Since storage systems must be made accessible to any given temporary system in this type of environment, any short-lived server or service that becomes compromised will have access to a persistent storage entity (or Tier) that is used by many other servers or services. This is often an effective way adversaries can laterally (or potentially, vertically) compromise any given network.

To restrict this, each server or service should have access to persistent storage, but should store information based on the application it represents, and, more importantly, the *unique Endpoint, Partner, or User* that the application is acting on behalf of. The last part of this point is the most essential point, as enforcing persistent storage access on behalf of a given *identity* limits the short-lived server or service's access to data.

In other words, an adversary that has compromised *short-lived system* can only affect the data stored on behalf of the identity tied to *that same short-lived system*. If that system only has access to a single identity's data, the adversary cannot use this system's compromise to migrate laterally to other accounts. They are restricted to accessing information for that single identity. This significantly limits the adversary's ability to leverage a vulnerability into a significant exploit of the system.

5.5.1 Risk

If a secure persistent storage model isn't defined, there will be no architecture that enforces unique per-user attributes to be securely separated from other assets. The result can be that any compromise of a token that grants an adversary access to a storage device may result in the compromise of multiple user's data. A persistent storage model, however, can isolate the compromise to one single user, or a single storage technology with encrypted data. In either case, the scope of the compromise is significantly reduced, granting the organization more time to react and combat the threat to both users and the business.

5.6 Define an Administration Model

Each system must be accessible by administration to troubleshoot and diagnose application faults. This can be challenging in environments where services or servers are short-lived, if an administrative model is not sufficiently designed.

To accomplish this, identify how the administrative team will communicate with each system in each tier. There should be authentication boundaries, such as VPNs, that separate disparate systems from each other. Ensure the administrative team must authenticate through each tier.

Also, identify how the administrator will interact with the system. Can the system be snapshotted, akin to a VM? Is a terminal used? Is a remote Secure Shell (SSH) used to interact with the system? Are there APIs for monitoring and analysis of system metrics, such as CPU usage, disk usage, and network usage? Can those be used to troubleshoot or identify anomalies?

Regardless of the model, there are certain things that *must* be defined:

- How administrators will authenticate to the environment
- How administrators authenticating can be attributed to physical identities:
    - Using two-factor authentication (2FA)
- How systems snapshots can be taken
- How changes can be made and must be tracked

5.6.1 Risk

Environments without a well architected path for administrative access typically end up using ad-hoc means to access systems in production. This often leads to administrative ports that are open to public connectivity, or services that offer diagnostics, but are not restricted from being used by third parties. A clear administrative model reduces the potential avenues attackers can take to gain privileged access to critical IoT resources.

5.7 Define a Systems Logging and Monitoring Approach

Each system must be monitored to allow administrators and Information Technology (IT) works to detect and diagnose anomalies. Monitoring must be performed at multiple dimensions. For example, network monitoring at the infrastructure level helps diagnose application attacks or DDoS against network components. Tier monitoring identifies whether specific applications or pieces of infrastructure may have been breached. While system-level monitoring defines whether individual applications, or application platforms, are being attacked or have been compromised.

This obviously takes multiple levels of monitoring and consolidates the information into a resource that can be streamed to an oversight team. There are several professional applications that provide this technology, and convert the metrics into visual systems usable for IT professionals and system engineers.

Anomalies that indicate adversarial behaviour may include, but are not limited to:

- Increased network traffic
- Increased network traffic in an *odd* direction (especially egress)
- Egress network traffic from a resource that shouldn’t need egress
- Abnormal CPU utilization
- GPU utilization for systems with no visual interface, but have a GPU as a part of the CPU
- Disk or network-storage utilization
- Abnormal changes in system time on a particular host

While monitoring systems for capturing anomalies are readily available, the context may be specific to the application or infrastructure used by the organization. Consult with the business providing the monitoring system to determine how to capture and interpret metrics in a way that is most effective for the specific implementation.

Separate tiers may have differentials in anomalies indicating an attack or compromise. Evaluate what those indicators are per each tier.

Please consider using material from the following organizations to assist with this recommendation

- Amazon EC2 Monitoring Documentation
  - http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/monitoring_ec2.html
- Google Cloud Monitoring
  - https://cloud.google.com/monitoring/
- Microsoft Azure Monitoring
  - https://azure.microsoft.com/en-us/documentation/articles/best-practices-monitoring/
- DigitalOcean Monitoring Tutorials (General)
  - https://www.digitalocean.com/community/tags/monitoring?type=tutorials

5.7.1 Risk

Systems monitoring technology is a key attribute of the IoT security model. Without monitoring, there is no way to determine if a vulnerability has been found in critical service components. Monitoring allows administrators to quickly diagnose pain points in service and infrastructure, and can assist in differentiating between security incidents and software bugs.

5.8 Define an Incident Response Model

It isn’t enough to detect a potential compromise or an on-going attack. The organization must be able to react to, and combat, the attack. If a system is compromised, cleansing or shutting off that system is not enough. The organization must, instead, be able to diagnose the source of the compromise, patch the system, and deploy the patch on all existing infrastructure.

This may be difficult if a container-based environment is used where cloned applications are running with a vulnerable configuration. The application system must be able to detect a “reboot” or “update” event, where the application connection is handed off eloquently to another system in the Cloud, or the user is forcibly logged out, to permit the update to occur.

No matter what the execution model is, however, the engineering team must be able to capture metrics in a way that allows for forensic analysis. These policies and procedures must be set in stone and approved by the legal (and possibly the insurance team) to validate whether the information is contained in a way that is suitable for Law Enforcement Officers (LEO). Compliance will assist in ensuring the business is not only complying with local and federal law, but is providing samples of a compromise that can be used in court.

Once the samples are captured, each aspect of the overall system should be assessed for logs, metrics, and other data that can corroborate to the event in question. This data should all be captured and stored in a secure system for legal review.

Please consider using material from the following organizations to assist with this recommendation:

- CERT Recommendations for Creating a CSIRT
  - http://www.cert.org/incident-management/products-services/creating-a-csirt.cfm

5.8.1 Risk

Organizations that lack an incident response model will take far more time to organize their resources, identify compromised systems, quarantine those systems, and review the systems for information. This also significantly slows the efforts that should be made to patch and restore a given system. This unpreparedness grants adversaries a large window of opportunity to leverage one compromise into lateral or vertical movement within a given environment. This may result in a significantly larger compromise due to the increased response time. Organizations should be prepared to respond to an incident almost immediately to reduce the amount of time an adversary has to control critical portions of the service.

5.9 Define a Recovery Model

Regardless of whether a user or application is affected due to a security compromise or a hardware fault, a recovery must occur. A procedure should be put in place for recovery of information and capability within application tier. The procedure must be tailored to the context of each application and tier.

For example, if an application has gathered information from an Endpoint regarding the output of a particular action, and there is a storage fault that disallows the application from solidifying the output of that data in persistent storage, the application can:

- Attempt storage again until it succeeds (May be endless)
- Attempt storage for a limited number of attempts until success or failure thresholds
- Fail immediately, potentially losing the metrics
- Ask the Endpoint again for the same data (May never be available)

The method that is most suitable for the application and business requirement must be chosen. This, again, will depend on the context of the application, and may not be easy to model outside of a given system.

Engage both the engineering and business leadership to determine how a failed or compromised application should recover, especially in the context of user activity.

For systems that have been provably compromised by an adversary, there must be a model in place for validating that the application or system has been sufficiently patched prior to recovery. Without this policy and procedure set defined, a vulnerable system may simply be redeployed into the Service Ecosystem, facilitating further compromises.

5.9.1 Risk

Recovery models ensure that information, applications, and configurations are restored correctly. Without a restoration model, the team may unintentionally redeploy vulnerable subsystems to servers or infrastructure. Also, tainted data that could have been manipulated by an adversary in a database or storage environment could be replicated across multiple systems, unintentionally propagating malware or simply altered data. Recovery processes reduce the ability for adversaries to abuse weaknesses in the recovery of an incident, which adds to an already expensive event.

5.10 Define a Sunsetting Model

Every system that is deployed by an organization, and every tier used, has a lifetime. Even if the same product or service is deployed by the organization for decades, the technologies used to drive that product or service will change. Thus, there must not only be a plan for designing and implementing the product or service, there must be a plan to *sunset* that product or service.

This process helps guarantee that all technologies are revoked and decommissioned in such a way that an adversary cannot take on the identity of, or use the facilities of, the given technology. For example, a simple case is a domain for a particular product after a business's acquisition by a parent company. If the product is renamed, and the domain is migrated to the parent company's domain, an adversary may be able to take ownership of the now defunct domain. If the adversary can issue cryptographic certificates for that domain and still interact with deployed technology under that old domain, there will be a significant gap in security caused by a lack of procedure in the sunsetting of that product or service.

Each technology used in the architecture, implementation, and management of a given product or service must, then, be catalogued and evaluated for its usability. Once that technology is no longer usable, it can be sunsetted according to its model. This allows engineers and business leadership to migrate the technology over to a more suitable set of innovations without gaps in the underlying platforms. It also ensures that a product that will no longer be offered to partners and users will end its lifetime without the potential for exploitation by adversaries after the business closes down.

5.10.1 Risk

A lack of a sunset process can result in both Endpoints and services being compromised by competitors or adversaries. This is possible legally because if an organization releases access to certain objects, such as domain names, phone numbers, and other renewable services, an adversary or competitor has the right to acquire those objects, even if it seems unethical. This may open devices or services up to unscrupulous abuse or even malicious behaviour.

5.11 Define a Set of Security Classifications

To properly manage interactions with Partner organizations effectively, security classifications must be defined. This will set the tone for not only the internal organizational policy on data security, but will help define the level of security Partner organizations apply to the business's data, their own data, and customer's data.

While this process should be investigated and tailored to the organization, most data security classification policies should start out with the following classes:

- Public - Any entity granted access
- Classified - User must authorize release
- Secret - User-specific data
- Top Secret - Organization-specific data, never to be released

After defining the basic classes, the organization must evaluate how each security class should be attributed to a data class. In other words, evaluate how the classification should be used in practice, not just in theory. Determine what policies and procedures should be put in place from a business *and* engineering perspective.

This will allow the organization to not only build technical policy, but enact business policy that supports the technical requirements. This makes it easier for the engineering team to hand off these requirements onto partners and internal organizations that would seek to break the policy, either intentionally or unintentionally.

Once the security classifications have been standardized, it is important to evaluate how the security classification model can be affected by privacy requirements of the business and its users. The organization must take the time to apply a privacy model to the security classifications, to give meaning to users' data, and help protect their privacy in the event that a Partner wants access to specific resources that could put users at risk of exposure. By contextualizing privacy in the context of security classifications, Partners will need to seek approval by the business leadership *and* users, when Partners want to acquire certain types of privacy-centric data. The users *must* have the option to guard their privacy-centric data, and must be able to limit the exposure of their data to third parties.

5.11.1 Risk

Classifying models for security is imperative toward architecting solutions that use security in an effective manner. In order to protect information, that information must be quantified so that appropriate controls can be formulated based on corresponding policies and procedures. Without these models, engineers tend to implement security either too intensely, or not at all, depending on their perception of the risks involved. The entire team, including both engineers and business leaders, should identify what data means to the business and how it should be secured within an appropriate range of cost-effective controls.

5.12 Define Classifications for Sets of Data Types

After defining security classifications, the organization should define types of data to be used by the overall IoT product or service. This will enable the organization to clearly define what types of information are acquired, generated, and disseminated to peers in the IoT system, and how the organization should treat these types of data. This data will provide context and value to the overall components used throughout the IoT environment.

While this document will not attempt to model all variations of data that may be relevant to a specific organization, certain types may be as follows:

- Users
- Actions
- Images
- Editable documents
- Personally-Identifiable Information
- Protected Health Information

A piece of information may be attributed one or more *types*. But, the data itself should be attributed only one *security class*. While the *type* identifies what the data represents and how it should be processed, the *security class* will represent *how, where, and when* the information can be used, and *to whom* it may be shared.

Defining the various types of data and attributing classifications to them is a long-winded process. Doing so sets an organizational standard for the business and allows the engineering team to execute technical controls around the data and its classifications. This greatly assists the engineering and business leadership teams later when negotiating with Partners over how data can be shared and processed.

5.12.1 Risk

As with security classifications, controls cannot be implemented around data without quantifying what that data is, and what relationship that data has to the business. These classes define how the information should be used within the system, and what protections must be applied to the data in order to maintain an appropriate security posture. Without these classes, engineers have a tendency to apply too stringent or too weak of security measures. Security measures should be agreed upon by the engineering team and business leadership, to balance the controls with the importance of the data to the business.

6 High-Priority Recommendations

High priority recommendations represent the set of recommendations that should be implemented, but only if the endpoint architecture requires it. For example, not all endpoint architectures require tamper resistant product casing. These recommendations should be evaluated to determine if the business case deems them a requirement.

6.1 Define a Clear Authorization Model

While the privacy model deals with the way user's information is offered to Partners, the authorization model defines how the business or Partners will act on behalf of a user. This, for instance, would come in handy for a home automation system where a Partner's metrics could optimize the use of heating or cooling in a given home. The authorization model would grant the Partner the ability to change heating or cooling controls for that user's home when certain metrics were detected by the Partner.

To accomplish this, have a similar GUI that describes granular authorization capabilities, and how they will be distributed to Partners. Allow the user to grant access, or revoke access, to certain capabilities on demand. Ensure that revoked capabilities take action immediately, to diminish the potential for abuse.

The system must be heavily monitored to ensure that Partners do not take actions that they are not allowed to take. Granular control of the authorization model should allow users to configure when Partners have access to certain capabilities, and how often. Attributes like this will improve whether a user can take control of their system from a potentially abusive, or compromised (hacked), Partner.

6.1.1 Risk

Without an Authorization Model, third parties will not have restricted access to a user's capabilities. This may allow a rogue or compromised third party to acquire full access to a user's technology or data. By creating an authorization model, access is restricted to only the attributes that a user will allow. This enables the user to have greater control over what capabilities and data are made available to third parties, and reduces the risk of the IoT Service Provider by mitigating the potential for widespread compromise.

6.2 Manage the Cryptographic Architecture

All technology deployed in an IoT environment must use cryptography, regardless of whether the technology is a rudimentary low-power endpoint, or a robust Cloud service. To properly implement security in an IoT product or service, the cryptography used must be well architected, managed, and adjusted to meet changing specifications over time.

The engineering team must identify whether:

- Their cryptographic algorithms have been deprecated
- They are using cryptographic keys with adequate bit-lengths
- Hashing algorithms are subject to collision attacks
- A strong random number generator is used
- Messages are sufficiently padded with random data

- Cryptographic protocols, such as TLS, are up-to-date with best practices
- Privacy-centric concepts, such as forward secrecy, are used
- Plaintext passwords or pin numbers are passed over the network
- A custom cryptographic algorithm was used

Each of these points, and more, are important to maintain a high quality cryptographic architecture within the IoT product or service. Success in deploying a cryptographic solution is tightly bound to the engineering team's ability to leverage the most resilient cryptography solutions for deploying patches to technologies that use less resilient solutions.

For example, the RC4 algorithm was recently discovered to have significant security flaws. If a patch can be securely distributed to clients configured to use RC4, that replaces RC4 with AES-256, then RC4 becomes less of a concern. If mutual authentication is performed using a more resilient technology, such as Ephemeral Diffie Hellman key exchange and asymmetric keys, or a UICC security token, the patch can be verified without using the vulnerable cryptographic algorithm.

Passwords and pins used by a user or endpoint should never be passed over the network in plaintext, even if the communications channel is secured through encryption. Instead, the cryptographic hash of the password or pin should be used, to ensure that any misconfiguration in the cryptographic tunnel does not expose the password, itself. The hash should be generated by the password and at least one unique one-time token. While it is common for this token to be taken from the network session, it is more secure to take the value from a rolling code stored on both the endpoint and the service infrastructure. This way, an attacker with a position of privilege on the network cannot seed the hash with beneficial values, which may result in a forced-signing attack.

Custom cryptographic algorithms (algorithms designed in-house) should never be used. Always use recommended algorithms developed by cryptographers, and recommended by oversight organizations that specialized in cryptographic security. Always avoid the use of poorly designed algorithms, deprecated algorithms, or compression, binary-to-text, or other algorithms commonly mistaken for cryptographic algorithms, such as LZO, base64, ROT13 and XOR.

Please review the following guides and references for more information on this topic:

- ISO 18033-1:2015 – Encryption Algorithms
- ISO 18033-2:2015 – Asymmetric Ciphers
- ISO 18033-3:2015 – Block Ciphers
- www.owasp.org/index.php/Guide_to_Cryptography
- csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf
- csrc.nist.gov/groups/ST/toolkit/key_management.html

6.2.1 Risk

Properly deploying a solution with a cryptographic architecture ensures that the algorithms, protocols, and secrets utilized all fall within the current recommendations. In addition, the recommendations change over time. Without a cryptographic architecture it will be more difficult to identify all of the technologies that have deprecated, which creates an opportunity for gaps in security.

6.3 Define a Communications Model

Each system in the Service Ecosystem must be capable of mutual authentication. No computing platforms within this ecosystem should be accessible to anonymous public users. Each Endpoint, Partner, or User will communicate with the Service Ecosystem through technologies that require mutual authentication. Since the services that make up the user interface are typically deployed and managed in a separate environment, the publicly accessible interface must be confined to that space. The Service Ecosystem, however, comprises the set of all system used to deploy service to all authenticated resources.

This includes Endpoints that have not yet been provisioned by the system, as the hardware fabrication and personalization process *should* configure the hardware adequately enough that it may be *authenticated* as a business-deployed resource.

Therefore, the communication model *must* provide:

- Mutual authentication
- Confidentiality
- Integrity

To accomplish this effectively, the communications model must also provide:

- A centralized root of trust or, alternatively, a decentralized root of trust
- Identity provisioning and revocation
- Perfect Forward Secrecy

A root of trust must be used to ensure that each entity in the communications model is authorized by the same organization as the peer. This helps to ensure that all entities have been provisioned and authorized by one central organization. The technology used to ensure this root of trust may be centralized (similar to TLS certificates) or decentralized (similar to IoT models based on Bitcoin blockchain, for example IBM/Samsung’s ADEPT project, Tilepay, and others). Regardless, one central business must be the *owner* of the model, and guard the provisioning system.

Provisioning and revocation must be a part of the communications model, to help guarantee that any compromised secret or identity can be removed from the system with minimal effort. Technologies such as Online Certificate Security Protocol (OCSP) help with this process.

The communications protocol *must* employ a technology that mitigates the potential for compromising communications from *the past*. This is done by creating ephemeral asymmetric cryptographic keys that are used to exchange a communications secret. If a certificate is compromised, the ephemeral secret will not be. This ensures that storing encrypted messages for a long period of time will not later result in an adversary decrypting them if the certificate private secret is ever compromised or exposed.

The challenge with communications security is in the implementation and longevity of the technology. Encryption algorithms can be selected that are held in a high degree of confidence by authoritative entities, diminishing the potential for failure.

Libraries and algorithm implementations designed or approved by engineering authorities *must* be used. Custom implementations of algorithms *must not* be used. This diminishes not only the engineering team's work, but the potential for an algorithm to be cryptographically weakened by a poorly designed or incorrectly implemented system.

Please consider using material from the following organizations to assist with this recommendation:

- CafeSoft Apache Mutual Authentication How-To Guide:
  - http://www.cafesoft.com/products/cams/ps/docs32/admin/ConfiguringApache2ForSSLTLSMutualAuthentication.html

6.3.1 Risk

Communications security is the cornerstone of IoT. Without communications security, there is no guarantee that embedded devices are communicating with the correct back-end services. This is imperative for critical services that guide, configure, and send commands to devices such as telematics, medical devices, and industrial control systems. Without communications security, there are no guarantees that commands are being sent to the correct endpoint. Enforce communications security to ensure that messages are being sent to and received from the intended peer.

6.4 Use Network Authentication Services

Network Operators, when used as partners, allow users to be authenticated using tokens specific to the network operator. While these tokens, present in the Network Operator's UICC, authenticate a user to the network layer, they don't necessarily authenticate the user at the application layer. The use of the following technologies may facilitate network authentication:

- Generic Bootstrap Architecture (3GPP TS 33.220)
- M2M SM (ETSI TS 102 921)

Evaluate whether the authentication technology will create meaning at the application layer, for authentication purposes. If the token can be used as a security store, determine whether the device can be used as an authentication layer for the physical Endpoint to build a TCB using the token.

While many network operators enforce network-based authentication, granting access to this API to authenticate either users or Endpoints is a fairly new technology. Evaluate whether the Network Operator you are working with creates a meaningful experience in this space. If so, consider using this technology as more than a network layer authentication token, as it may be easier to utilize one security store technology, instead of multiple technologies.

6.4.1 Risk

When network authentication services incorporate trust anchors such as the UICC, *not* utilizing these services to secure the application layer will limit the ability for the application to reliably authenticate users and will increase the expense of the underlying Endpoint platform. This increases the cost of deployment and also decreases the information available to the organization from the Network Operator.

6.5 Provision Servers Where Possible

Server provisioning involves defining, configuring, personalizing, and deploying a server in a production environment. The provisioning process, from a service perspective, ensures that a server is security hardened and ready for deployment in an environment that may be potentially hostile.

Regardless of whether the server is deployed in a Cloud infrastructure, a dedicated hosting provider, or in a company’s personal rack space, a server will be vulnerable to both insider and external threats. The server must then be hardened from attack before it is ever deployed in the service infrastructure.

To accomplish this, identify the services that should be accessible to the surrounding environment. Define whether the environment the server will live in will be public or private, and what that means in the context of server security. Determine whether each service running on the server should be accessible to the public, or whether only authenticated clients must connect to the service.

Evaluate the lifecycle of the operating system that will run on the server. Determine how to properly manage software updates to ensure that security patches will be quickly deployed and committed to servers working in production. Evaluate a roll-back model in the event that updates fail or cause unexpected issues with production services, as some library or application updates may result in unintended side effects.

Finally, evaluate the sunsetting model of the provisioned server to determine the most secure way to remove assets from the system. This includes system logs that may be required for assessing anomalous service or client behaviour.

This recommendation implies that a Patch Management process should be implemented by the organization to identify vulnerable services, deploy patches, and monitor the success of implementing those patches.

Please review the following material on Patch Management:

- http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-40r3.pdf

6.5.1 Risk

Server provisioning is an imperative part of the overall security of an IoT environment. Without it, the organization’s control over the server architecture will be substantially weakened. This may result in gaps in security due to a lack of architecture specification. Without a specification, the organization cannot review whether the deployed technologies adhere to modern best practices. In addition, improving these technologies will require investigation of each deployed system to evaluate the deltas between deployed assets. This is inefficient and a high concern in the event that a critical security update needs to be deployed. If there is no consistency and no architecture to define the services, there will be no way to easily track which systems require immediate attention without manually checking each one.

6.6 Define an Update Model

Updating an execution environment, application image, or TCB is a challenging process. Consider the following example model that simplifies the overall process:

- For each layer of the execution platform, define a network resource such as a unique URL for the new application image
- Generate a signing key for each specific layer
- For all new, authorized versions of each layer, generate an image of that layer
- Include metadata describing the image (version, timestamp, identity, etc.) in the layer image
- Sign the layer image with the signing key
- Make the image, the signature, and the public key available, possibly via the unique network resource, or through a update service

When a new system is deployed it should:

- For each layer:
    - Retrieve the version(s) to be deployed
    - Cryptographically verify the image
    - Deploy the image layer on the system

No private secrets should be stored in any application layer. Instead, secrets must be provisioned dynamically as each system is deployed, to personalize each system. These identities should be revoked as the system is decommissioned, no matter what the lifetime length of that system is.

This recommendation implies that a patch management process should be used to maintain services and technologies within the infrastructure.

Please review the following documentation for more information:

- http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-40r3.pdf

6.6.1 Risk

Without a well-defined update model, services and applications risk compromise through abuse of the update procedure. Adversaries may be able to inject custom applications into the update process and deploy their own software onto Cloud systems and other servers. If the communications security infrastructure is not secured, this can easily be performed simply by manipulating network services such as Domain Name Service (DNS). More advanced attacks against routing, such as Border Gateway Protocol (BGP) attacks, have been implemented many times in the past to compromise insecure services.

6.7 Define a Breach Policy for Exposed Data

Defining policies and procedures for the classification of data is not enough. There must also be a model for detecting whether the data has been exposed by a Partner. The organization must have a plan in place to evaluate whether a Partner was involved in business practices that breach the technological controls or policies set in place to guard user's data and privacy.

To accomplish this, the engineering team must define monitoring and logging technologies that apply to *security classifications* and not simply user data. This will allow an auditing trail to apply not only to information, but to the *classification* of that information. This will help the organization defend itself in the event that user information is exposed. The organization will be able to show that its security classifications, and the technical controls put in place to manage those classes, managed, stored, and disseminated the data according to policy.

It is beneficial that the organization use the monitoring and logging technology to prove when a Partner has broken the rules of the security classifications. The leadership should, at that point, decide whether or not the Partner should be subject to fines, dismissal, or another consequence.

6.7.1 Risk

Without a breach policy, there are few legal guards to protect the organization from liability of data that has been exposed by a third party. If the business is the source for the exposed data, the third party may have lost the data, but the business is liable for the data it hands off to its partners.

Breach policies ensure that partners must uphold a level of security adequate for the data that they are being provided. Any breach of that security helps to removes the liability from the IoT Service Provider as long as the IoT Service Provider follows its own security requirements. Then, it is up to the partner to adhere to the policy.

These policies should be reviewed by legal and insurance teams to ensure that the models do, in fact, reduce the liability of the organization by adhering to stringent security policies and procedures. Some businesses, due to the nature of the products or services they offer, may not be exempt due to regulations, legal statutes, or other issues.

6.8 Force Authentication Through the Service Ecosystem

A user interface should *never* authenticate a user directly. The system must always be able to authenticate the user by using the centrally available service. The only exception to this rule is if an application running on a mobile device is guarded by a local passcode. This passcode may be used to access the local application. However, access to remote services and resources should be verified by a separate authentication token.

Although, for usability, the engineering team may choose to collapse these two authentication schemes into one if the user is provided with sufficient information that describes to them the risks in utilizing this method of authentication. Such a method would enable an authenticated user's local application password to decrypt a local database containing an authentication token that works on the remote service. This multi-step authentication model may be sufficient for most users.

Regardless, the central Authentication Service must first authenticate the user to the local application, then enforce policies and procedures that mandate how that authentication token can be used, and for what period of time. Metrics should also be gathered to determine if the user has migrated to an alternative computing platform, but is using the same token. Or, if the user has migrated to another location in a short period of time, but is using the same token. Depending on the type and speed of movement, these metrics may indicate a potential compromise of the token. At which point, the token should be invalidated, and the user should be forced to log back in, potentially by using multi-factor authentication, where applicable.

6.8.1 Risk

Due to the abuses possible in Endpoint systems, regardless of how secure the architecture can be, authenticating a user without confirmation from a back-end system is always unreliable. This presumes that the user has not updated their credentials, or can fragment credentials across multiple types of devices. This is inefficient and may open up a gap where a compromised device is using an older version of the user's credentials.

6.9 Implement Input Validation

All data acquired from an endpoint, user, or purported user, must be analysed for anomalies. The easiest route of attack for an adversary is always abusing web application input in the services that make up the user interface. This is because this technology must render information dynamically, based on the variations in locality, encodings, and other parameters that change from user to user. Skilled users can manipulate certain attributes of encodings to cause unexpected side effects beneficial to an adversary at different layers of the processing subsystems.

For example, a fascinating attack includes the encoding of a null byte into messages that are processed as strings by higher level languages. Some high level languages accept null bytes as part of a binary string, rather than seeing it as a delimiter. When this binary string is passed down to lower level libraries, the embedded null byte is interpreted as a string delimiter, truncating the string to mean something entirely different than what the application interpreted the string as. In the past, this has been a clever way to access file system resources that would otherwise be unavailable to a particular user.

While there is an infinite number of variations for malicious input to take form, engineers do not need to test for every possible case. Instead, the process is fairly simple:

- Identify how the data shall be used internally
- Enforce a policy around what kinds of encodings and characters adhere to the internal use model
- Design an API that analyses the data according to this policy
- Raise an exception when data has been identified that breaks the model
- Log the event internally with metadata regarding the session to help detect adversarial behaviour

All data stored within the system should first be processed and composed into a static model. An effective technique for this is simply encoding all data with the base64 algorithm, then placing it in a database. This ensures that the data can in no way manipulate the database.

6.9.1 Risk

Systems that do not employ input validation are subject to an array of possible attacks including issues referenced in the OWASP Top Ten such as SQL Injection (SQLi), and even remote code execution attacks. Because the range of potential abuses is so wide, the risk cannot be fully quantified here. Input validation is a critical attribute of any secure application, whether it is a cloud service or an application running on an Endpoint.

6.10 Implement Output Filtering

Output filtering is the complement to input validation. This process not only secures the presentation layer from adversarial manipulation, but also disallows the system from rending information to a user that should be deemed privileged.

In the first case, all data to be rendered by the presentation layer must be assessed prior to it leaving the service layer. This will ensure that data encoded into the presentation layer, for example, in JSON messages or encoded JavaScript, does not contain formatting that could break or invalidate the presentation of the data. This means that any characters stored in the system that, if rendered, could break the presentation model must either be filtered out or encoded in such a way that they do not alter the presentation in unexpected ways.

A methodology for remediating this issue is either filtering restricted characters out, enforcing an encoding on all characters so that the presentation of these characters does not alter the GUI (the characters aren’t interpreted by a rendering engine as control codes), or simply not displaying the message. While any of these methods work, some are more appropriate in certain applications. Reviewing the message forum example, it would be just as bad if an adversary were able to place scripts that other users can copy and execute without knowing what they are doing. Therefore, instead of just rendering the information in a way that doesn’t inject HTML or other scripts into the presentation layer, the information should be scrubbed so that other users are not affected.

In the case where data should not be presented back to the user, this does not relate to data stored and rendered by an adversary. Rather, this issue relates to the manifestation of data that isn’t suitable for public consumption, and should be reserved for administrators and engineers. For example, if an internal error is generated in processing information, that error should not be rendered, with its full debug data, to the user. This may allow a user to identify and instrument a bug for the purposes of exploiting weaknesses in the application. This information should be internally logged, and a generic error should be raised to the user that does not provide enough context for the user to abuse the bug. Even if the user can reproduce the bug, the user should not be able to evaluate a differential in output from the application that indicate improvements in the exploit methodology.

6.10.1 Risk

Output validation is a critical attribute of IoT security. Systems that do not perform output validation risk exposure of critical user data, privacy related data, diagnostic data, verbose error messages, and more. These messages may be used to expose user information or can be used to create a reliable exploit against a networked service.

6.11 Enforce Strong Password Policy

It is imperative that all authentication systems enforce strong passwords where passwords are required for user authentication. Password complexity has been a constant battle between information security researchers, engineers, and business leadership. Business leaders often want users to be able to remember their passwords easily. Engineers need to reduce the complexity of interfaces, especially for designers of the presentation layer. Information security researchers often overestimate the skill of an attacker and press unnecessary complexity onto a given technology.

The answer, however, is somewhere in between all the requirements of each group. Passwords should be forcibly long, but should not be complex. While eight character passwords used to be the norm, and some systems even allow 6 characters even at the time this document is being written, the password length should be taken from the latest best practice standard, but will likely far exceed even 8 characters. By enforcing a longer password length, the complexity requirement is reduced. Instead of enforcing bizarre assortments of various character sets, the user can simply remember a sentence. Because they can choose to utilize white spaces, capitalization, numbers, and punctuation, the complexity is automatically skyrocketed for any attacker applying brute-forcing.

Let’s not forget, there are typically four ways an attacker will compromise a password:

- By stealing the password database and cracking individual passwords
- By brute-forcing the application authentication service
- By installing malware
- By using hard-coded or default passwords

Forcing long passwords helps diminish the first risk. But, security at the Service Ecosystem layer is far more beneficial. The attacker shouldn’t be able to retrieve the password database in the first place, which brings us to the second point.

Brute-forcing application passwords is then the most effective way an attacker can abuse passwords. This potential is significantly diminished by a properly designed authentication service. If one bad password has been guessed, the system should automatically start increasing the amount of delay required between guesses. Then, a threshold must be defined that limits the total number of attempts. If the attacker reaches this threshold, the account should be locked, and two-factor authentication, or another model, should be used for the user to unlock and verify their account. This type of security substantially reduces the benefit of a network based attack, which brings us to the final point.

Malware on the client system is something that must be dealt with by the computing platform, or by the user installing the appropriate combative technology. This is typically not something that can be secured by the application, itself. Since there is little to nothing the application can do to combat this risk, beyond enforcing 2FA, the application engineer will have satisfactorily reduced the threat surface of password attacks against the authentication system if this is an adversary’s *only viable course of action*.

It must be noted, however, that the *reward* for implementing this recommendation is *not high*. This is because no matter what technologies are used to reduce the potential for attacking password authentication, passwords are, essentially, an intangible resource. They are not physical tokens that can only be captured by a single individual. Rather, it is an abstract object that can be copied infinitely across computer systems and through visual observation. Therefore, they are a significantly weak source of authentication that does not, in any way, adequately indicate a particular user. Therefore, passwords, themselves, are a weakness, and any technology using passwords is subject to the risks that passwords inherently imply.

Passwords should never be hard-coded in the system. For endpoints, unique cryptographic keys should be generated. Refer to the Endpoint document for more information on endpoint provisioning. For services and user interfaces, the password should be defined by the user when they register. The password, at that time, must adhere to strong password security requirements. Never allow a user to utilize a default, weak, or poorly designed password.

Ensure that the user always has the ability to change their password at any time. Enforce strong authentication requirements and communications security in order for the user to change their password. Where possible, enable two-factor authentication (2FA) to verify the identity of the user prior to allowing a password change. Always force a user to re-enter their original password when they submit a new password to the system. This ensures that another user hasn’t usurped an open web application by taking advantage of an unlocked laptop, or stolen a web application session token.

6.11.1 Risk

Systems that don’t enforce adequate password controls risk the ability for adversaries to easily guess passwords from users of the system.

6.12 Define Application Layer Authentication and Authorization

While the Organizational Root of Trust and its services will define authentication technologies that secure the network communication layer, the user, administration, and partner authorization technologies must be configured separately. While these entities' communications channels are secured with the Organizational Root of Trust, their *actions* and *identities* must be authenticated using a separate system.

Generally, this application layer authentication will be facilitated by the same service. However, information will be gathered from a separate resource. For example, it is best to place user and administrative authentication data in separate databases. This ensures that if there is a way to manipulate the database through the application layer (for example, using a SQL injection), attackers may only move laterally through the user database. They may not move vertically, elevating their privileges to administrator, without compromising the database, itself. This is a significant improvement in organizational security.

If possible, define separate storage systems for:

- Endpoint identities
- Users
- Administrator credentials
- Partners

This will create a logical separation of duties for applications and infrastructure, but within the same authentication API managed by the Organizational Root of Trust service.

Please consider using material from the following organizations to assist with this recommendation:

- OAuth 2.0 [8]
- OpenID Foundation [9]
- GSMA Mobile Connect [10]

6.12.1 Risk

Without a methodology for enforcing application layer authentication and authorization, there is no way for the system to confirm the actions purported to be from a user are actually authorized by that user. Implementing this recommendation ensures that each action is traceable to an authenticated user and an authorization. These metrics can be stored and later reviewed in the event that a compromise is suspected. Without these steps, there will be no safeguards that minimize the risk of abuse.

6.13 Default-Open or Fail-Open Firewall Rules and System Hardening

In some service infrastructure environments, ingress and egress protection mechanisms are not configured by default. This means that engineers must employ firewall or network traffic rulesets themselves. These rules must be set in infrastructure before any service is deployed to the public.

Yet, there are times when these technologies may not be sufficient in protecting service infrastructure. Sometimes, firewalls and other network traffic protection systems fail. When these systems fail, they often fail open. The reason for this is that if the system fails, traffic must still be able to work, as traffic for other computing environments will be routed through the infrastructure along with the IoT Service Provider's traffic. Thus, traffic cannot suddenly come to a stand-still. As a result, the system often fails open to allow as many services as possible to continue working.

The engineering team should employ operating system hardening to ensure that the effects of failed infrastructure do not result in a catastrophic security event. Instead, it simply means that more connections can be made to existing service infrastructure.

For example, hidden services should not be placed behind technology such as firewalls. Instead, Virtual Private Networks (VPNs) or other high-security protections can be used to secure the services from adversaries.

Note that software firewalls carry an additional risk, in that they can be manipulated by a savvy attacker. If a software firewall is used, any server infrastructure that is improperly hardened may be manipulated by an attacker. In other words, if a public service running on a server carries unnecessary privileges (such as super-user privileges) and is compromised, the attacker will likely be capable of disabling the software firewall. Thus, the engineering team must evaluate whether a software firewall is too high of a risk for the chosen architecture.

6.13.1 Risk

Without employing strategies to compensate for failure in the network traffic security systems, the environment will be subject to unnecessary attacks that could be easily prevented with standard service hardening strategies.

6.14 Evaluate the Communications Privacy Model

Communications privacy is a slightly different topic than application privacy (described above) or communications information security. While privacy is largely evaluated from the ability for third parties to *effectively read* or *intercept* data, confidentiality and integrity do not represent the full scope of communications privacy.

Other issues that affect communications privacy include:

- Cryptographic uniqueness of each message
- Transmission patterns
- Plaintext metadata
- Hardware addresses or attributable serial numbers

While each message should be confidential and have verifiable integrity, they must also be cryptographically unique. If certain messages are sent in response to events that are predictable by an adversary, any responses that aren't cryptographically unique may be replayed by the attacker. Each message should be unique to disallow the attacker's ability to capture and replay messages that are beneficial.

Patterns in transmission can allow an adversary to identify a particular user, or equate behaviour with a certain attributable action. For example, technology that emits a message when a user enters a certain physical zone may be fingerprinted by “sniffers” that are capable of receiving these messages as they are transmitted through the air. While it may not be intuitive, this is a potential liability if an adversary can identify who is in a physical location and where they are in that location. Network patterns should be assessed to determine if there is a simple way that adversaries can turn transmission patterns into actionable data.

Metadata has long been used by intelligence services to evaluate the context of messaging systems without requiring a warrant or other legal access to encrypted data. Often, the metadata is enough information for an organization to create actionable intelligence. However, now hobbyists, criminal organizations, and curious users are able to use metadata for tracking and other potentially nefarious purposes. As a result, it is more important than ever to diminish the amount of metadata available to third parties. Where possible, limit the amount of metadata to only enough information required for a communication peer to evaluate whether the message is intended for them.

Along that line of thought, the hardware address of the communication module, and any unique serial numbers, should be protected or randomized, if possible. For example, Apple changed the iOS model for probing Wi-Fi access points. Instead of using the static hardware address, they changed their technology to use a randomized hardware address, which diminishes the potential for someone to track a user’s location based on Wi-Fi active scans. IoT technology will function similarly, but will have a larger set of communications technologies affected by this issue. Some technologies will not be capable of random hardware address generation, such as cellular. But others, such as 802.15.4, Wi-Fi, and Bluetooth, may be capable of this depending on the firmware’s functionality.

6.14.1 Risk

While it should go without saying that communications security is a requirement, it is sometimes confusing as to *why* it is a requirement. Communications security doesn’t just ensure that an adversary can’t read data. It also ensures:

- An Endpoint cannot be impersonated
- A critical service cannot be impersonated
- Abused messages can be detected
- Changes to software or security configurations can be performed safely

Without communications security, there are no guarantees as to the quality, reliability, or privacy of an IoT product or service.

7 Medium-Priority Recommendations

The medium-priority set of recommendations encompasses the set of recommendations that are relevant depending on the design choices of the endpoint technology. For example, enforcing operating system level security enhancements is only valid if there is an operating system running on the endpoint. If the endpoint is composed of a monolithic kernel application, or an embedded Real-Time Operating System (RTOS) with a single embedded application, the recommendation may not apply. Where recommendations *do* apply to the endpoint design, they should be implemented.

7.1 Define an Application Execution Environment

Several points must be made about application execution environments:

- The programming language used may have a direct relationship to security:
  - Languages like PHP and Ruby may present security concerns
  - Languages like GoLang and Erlang may decrease risk
- Third party libraries must be monitored, managed, and audited for risk:
  - Some libraries aren't well maintained
  - Some libraries have never been audited for security flaws
  - Some libraries require out-of-date dependencies that have known security flaws
- Always run an application as a non-privileged user:
  - If the application requires a privileged resource, use a wrapper to provision that resource before dropping privileges and executing the full application
- Use a well-defined TCB and Bootstrap model:
  - Applications that have well defined environments are more *reliable* and more *secure*

Please consider using material from the following organizations to assist with this recommendation:

- OWASP [5]

7.1.1 Risk

Applications that are deployed with a secure architecture can be subject to compromises that cannot easily be traced back to a specific source. Tools and techniques for compromising services and applications have become advanced in the past decade. Some open source technologies, such as Metasploit, allow the development and integration of custom exploits into an attack platform that can provide technologies to increase the stealth of an attack.

A secure Application Execution Environment can combat this risk by securing the way applications are ran, interact with each other, and the types of technologies that are used during runtime. These attributes can not only lessen the likelihood of a compromise occurring, they can add traceability and critical logging capabilities to track and diagnose the abused vulnerability.

7.2 Use Partner-Enhanced Monitoring Services

If the Partner being utilized is a Mobile Network Operator, identify whether they are able to offer monitoring services. Some network operators are capable of analysing the behaviour of endpoints communicating through their network. Operators with this type of capability have experience evaluating what metrics equate to anomalous and adversarial behaviour.

This will allow the IoT business to more quickly identify whether a particular user or Endpoint is either a threat, or has been compromised by an adversary. As a result, businesses may react more effectively to pre-empt attacks against other areas of the business's infrastructure.

The complexity with this service comes from the network operator's ability to provide the intelligence in a meaningful timeline. If the network operator can only provide the intelligence once the adversary has attacked the IoT business, then monitoring and logging systems placed in the IoT business's infrastructure should be able to detect the behaviour. However, if the network operator is able to notify the business of adversarial behaviour at the network layer, and can identify which individual subscriber has been emitting anomalous network traffic, the business may be able to limit exposure of the IoT ecosystem by cordoning off that single user's traffic.

7.2.1 Risk

There are certain technologies that the IoT Service Provider will rely on that cannot be monitored by the IoT Service Provider. One such technology is the communications network that connects an Endpoint to the Service and Network Ecosystem. Without monitoring services, the IoT Service Provider will not have a window into the events occurring within the network, itself. Thus, if an application-level identity A is attempting to compromise a service, the organization will be unable to identify that Endpoint B is actually the unit that has connected to the communications network. This gap in information is critical, as the organization may attribute the attack to identity A rather than a compromised Endpoint B.

7.3 Use a Private APN for Cellular Connectivity

An Access Point Name (APN) is a cellular communications component that connects the wireless network to the Internet. This point acts as, essentially, a Virtual Private Network (VPN) between cellular endpoint equipment and service infrastructure that the endpoint must interact with. A Private APN (sometimes called a Secure APN) is a version of an APN that has been security hardened to implement several desirable controls:

- Limited access restricted to authenticated clients
- Firewalling
- Endpoint-to-endpoint communication is forcibly disabled
- Monitoring services for anomaly detection
- Optional security or monitoring services

By restricting access to the APN, an organization can ensure that only authenticated endpoints are allowed to connect to the service infrastructure made available through the

APN. This diminishes the potential for rogue or random wireless clients to connect to the APN and access restricted services. In addition, it allows the organization to identify which specific clients are behaving anomalously, which allows the organization to tie negative behaviour to a specific piece of hardware, or a specific user.

Firewalling ensures that entities attached to the APN from both the client side (Endpoint Ecosystem) and the service side (Service Ecosystem) are restricted from communicating by using unapproved channels. This also restricts the ability for an Endpoint to abuse the APN as a channel to the open Internet, and cordons off traffic to a specific set of approved services.

Endpoint communications restrictions ensure that rogue endpoints are unable to attack other endpoints by using the APN as a wide area network. Instead, all communication must pivot through services approved by the organization. If desired, the organization can disallow point-to-point communication entirely.

Monitoring services enhance the security improvements that will be made by the organization in monitoring existing cloud or service infrastructure. By pairing those existing monitoring services with the APN and network monitoring technologies offered by the Network Operator, the organization can more easily track down the source of anomalous behaviour. This enables the organization to more deeply inspect incidents that occur against its endpoint or service infrastructure. For example, if the application layer indicates that User A may be compromised, but User B’s equipment is making the authenticated connection to the APN, the organization will be able to use the APN monitoring services to identify that User B has potentially compromised User A, or an adversary has compromised both User A and B.

Network Operators have additional services that can be layered on top of the services described above. These services will help black-list bad actors in the network, monitor specific users or sets of users, and can reroute certain types of traffic that may indicate anomalies. Other options may be available. Engage the Network Operator to determine which services are right for your organization.

While utilizing all of these services in concert may seem challenging, working with the Network Operator will simplify the process, and make it simpler to integrate these offerings into the business’s existing infrastructure. The complexity will come from utilizing the data effectively, and requires an engineering team that is capable of processing and managing the data in a reasonable fashion. Some services may incur an additional cost. Determine what pricing model and services will work best for your organization.

7.3.1 Risk

Without a Private APN, an Endpoint device can connect to almost any service or technology, including making direct connections to other Endpoints on the APN, or arbitrary service on the Internet. Since this would allow a compromised Endpoint to interact with almost any service on the Internet, and could turn the Endpoint into a practical target for acting as a proxy for attacking more secure networks or services, this recommendation should be enforced to restrict the ability for Endpoints to make arbitrary, unauthorized connections. It is much more valuable to the business and to the security of the entire IoT ecosystem when Endpoints are forced to connect only to approved services.

7.4 Define a Third-Party Data Distribution Policy

After security classifications have been defined, and data types have been attributed a valid classification, and a breach policy has been enacted, a data distribution policy should be generated. A data distribution policy describes how information should be processed through technical controls and out to service applications that have been granted permission to access the data. The permissions model is a part of the data distribution policy, and pairs with the user’s ability to create granular data permissions.

While a data distribution policy can be highly descriptive, there are several key elements that will help define a successful policy:

- What level of mutual authentication is required to traffic this data
- What confidentiality and integrity of data is required
- What ability does the business have to retain the data
- What ability does the Partner have to retain the data
- If retention is allowed, what period of time may the data be retained for
- What level of storage security must be applied to the data
- What access security classification must be applied to the data

7.4.1 Risk

Data distribution policies enforce security requirements on partners who may not adhere to the same level of security internally as the IoT Service Provider does. Since the IoT Service Provider cannot control the security a partner has implemented in their internal services and network, the IoT Service Provider can only enforce that data contributed to a partner is trafficked in a secure manner. Without this definition, the partner can enforce insecure configurations that may expose user data to adversaries while the data is still under the control of the IoT Service Provider. By enforcing stringent security controls for the communications channel, the IoT Service Provider proves it is doing everything it can do to enforce security until the data is out of its control.

7.5 Build a Third-Party Data Filter

Accepting dynamically generated data, such as advertisements, from a Partner requires a certain level of presumption regarding the quality and security of the data. Instead of making presumptions and applying the data to the presentation layer, the engineering team must take steps to ensure that the data distributed from the service application to or from a partner is well formed and does not contain potentially malicious content.

To do this, the engineering team should consider the following model:

- Does the data fit the format the Partner outlined for the data model
- Is the data well-formed
- Does the data represent a polymorphic object that could be misinterpreted by the client
- Will the data affect the way the client renders the presentation layer

- Will the data affect how the client *interprets* the presentation layer
- Does the data entice or request the user to perform a behaviour that would weaken security
- Does the data spoof or impersonate a component (password input field) of the client GUI

Reject any data that doesn’t fit an approved model. Notify administration immediately on detection of such data, and include as many metrics as possible regarding the origin and format of the data. Log a sample, if possible, in a secure database.

7.5.1 Risk

Dynamically generated data from third parties could contain malware, inappropriate content, or other undesirable data, either intentionally or unintentionally. Without an ingress filter shaped toward the definition of the third party service, the organization can risk accidentally allowing malware or other malicious content to reach the end user. This may result in system compromises, or simply lost customers, due to the side effects of such data.

8 Low-Priority Recommendations

Low priority recommendations encompass the set of recommendations that apply to risks that are extremely costly to combat, or are unlikely to affect the endpoint design. While these recommendations are valuable, and the information detailed within the recommendations is important, the mitigation or remediation strategies discussed may be out of scope with respect to the business. Evaluate each recommendation and determine whether the risks described are relevant or important to the business and its customers. If the customers require these risks to be addressed, apply the recommendations.

8.1 Rowhammer and Similar Attacks

Some implementations of modern RAM technology such as Dynamic Random Access Memory (DRAM) and Static Random Access Memory (SRAM) are vulnerable to errors that can be provably induced by certain memory access sequences. Abusing this type of error can result in the alteration of a specific bit, or bits, in predictable areas of memory. A successful exploit of this condition can alter bits in memory that represent types of privilege denoted by software.

In other words, if exploited correctly, an adversary can elevate their privileges from one user to another user by manipulating a hardware flaw in modern implementations of DRAM or SRAM. Many modern implementations of DRAM and SRAM have been found provably exploitable through this vulnerability. However, it requires the ability to execute code on the local system in order to create the memory access sequences capable of triggering this bug.

And yet, it may be possible to trigger this type of behaviour remotely through runtime languages such as sandboxed GoLang, Python, Erlang, and more. However, the preciseness of these types of attacks has not yet been documented, and is highly improbable to work effectively as an exploit.

This attack must be resolved at the hardware level. However, engineers can mitigate the risk of abuse by disallowing clients from executing code, even through a virtual machine or runtime, on a given service. By restricting this capability, engineers will be able to stop adversaries from creating the memory access sequences that are required in this attack.

8.1.1 Risk

Without sufficient guards against this kind of attack, adversaries may be able to remotely elevate privilege or execute arbitrary code against a target host. It should be noted, however, that a successful attack requires an extremely deep knowledge of the hardware, operating system, attack vector, and other factors that make this attack unlikely and rare.

8.2 Virtual Machine Compromises

Modern service infrastructure often utilizes virtual machines to deploy services on demand. While this model has proved extremely convenient and easy to deploy with, the problem with this methodology is the security of the overall infrastructure. While the engineering team may succeed in deploying a well thought-out architecture, the organization that manages and deploys the virtual infrastructure may not be as successful.

One major concern of deploying in virtual server environments is the ability for hosts to be compromised, or for servers (virtual guests) to intercept the data of other guests running on the same infrastructure.

While these attacks are valid concerns that should be evaluated by the IoT Service Provider, they often take a large amount of skill and time to perfect. Thus, it is a possibility that an attack may occur, but it will likely be a rare event. However, if the service infrastructure isn't well guarded, it will be possible that adversaries may be able to compromise administrative access to the virtual machines. This kind of a breach may not require a high amount of skill to succeed.

One way to combat this issue is with server provisioning. This process will ensure that each server is encoded with a unique set of cryptographic keys. If this process is followed, any compromise to a single server can be limit to that single server.

8.2.1 Risk

The risk of not accommodating for this type of attack may leave the service infrastructure vulnerable to many types of attacks. Server impersonation using keys accessible from the service infrastructure, data exfiltration, privacy compromise, and user impersonation may be possible.

8.3 Build an API for Users to Control Privacy Attributes

All users must be able to control what information they offer to third parties, through service APIs. The information should be classified into types of data, and attributed with security classifications. Users should be able to retrieve the types of data and classifications that are used in the modelling of their account. The user should be able to apply constraints to the types of data, to allow them to grant or revoke access to this data to Partners.

This can come in the form of an authenticated API, or a GUI that allows simple Yes or No controls on a general, and per-Partner basis.

8.3.1 Risk

Without the ability for users to control what data they contribute to an IoT Service Provider, they run the risk of their data being exposed in the event of a security breach either at the Service Provider, or one of the partners the Service Provider uses. Since certain users are at much higher risk than others, each user should be able to tune their privacy restrictions according to their personal needs. Making this interface available helps to ensure that the capability is there. The user must take it upon themselves to adjust the controls to suit their needs. For example, oneM2M (through TS-0003) allows the user to set the privacy preferences for a service provider.

8.4 Define a False Negative/Positive Assessment Model

While false positive analysis is an extremely complex topic, there is a simple way to identify whether a technology is more likely to present false positives. This is by evaluating the following items:

- Is the data source *trustworthy*
- Can the data source be tampered with or spoofed

- Is the data source from the analogue domain
- Can the data be corroborated from multiple points of origin
- Do the corroborating data sources exist on the same endpoint system
- Are corroborating data sources easy to tamper with or spoof
- Are tools readily available to manipulate the data source
- What level of expertise or cost is required to manipulate the data source
- Is the device attached to the data source *trustworthy*

All of these attributes, and more, can be used to evaluate whether data is *trustworthy*. This is extremely important, as critical decisions that affect the physical world can result in potentially harmful effects. It is imperative that the engineering team create a model for trustworthiness and apply it to each data source involved in making critical decisions. If the weight of the data source is such that it cannot be trusted, the most rational, and safest, action should be taken.

It is important to note that the engineering team is *not* the only entity that must make this kind of decision. The business leadership, the lawyer team, and the insurance team should be involved in the determining the correct reaction in potentially dangerous scenarios. Engineers must then encode the correct decision making process into the technology in a *verifiable and reproducible way*.

This process is highly challenging as it demands the attention of the entire organization on how the technology should react in critical scenarios. Trustworthiness is a challenging attribute to apply to a piece of technology, especially an embedded technology.

8.4.1 Risk

Without an assessment model for false positives, engineers may spend too much time analysing benign events while more important events are occurring. This may result in increased risk of the metrics analysed by the organization do not provide clear direction as to what types of events are occurring in production. This devalues the logging and monitoring infrastructure, and negates the organization’s ability to use these expensive resources to its benefit.

9 Summary

In summary, almost every security risk in an IoT product or service can be combatted by a well-defined architecture, intelligence to identify risks before and during security related events, and policies and procedures to handle such events. By analysing which high-level security concepts are important to the IoT Service Provider, frequently asked security questions can be reviewed. This should guide the engineering team toward which recommendations are most relevant to resolve gaps in their security architecture.

As the team progresses in its architectural definition, it can review standalone recommendations as their security questions and concerns become more unique to their own implementation.

Overall, every engineering team will face very similar risks. It is imperative that the organization choose to share their concerns with their peers to build common a knowledgebase for both risks and remediation strategies. Together, our organizations can build both technology and knowledge to assist each other in building security into the future of IoT.

Annex A Document Management

A.1 Document History

Version	Date	Brief Description of Change	Approval Authority	Editor / Company
1.0	08-Feb-2016	New PRD CLP.12	PSMC	Ian Smith GSMA & Don A. Bailey Lab Mouse Security
1.1	07-Nov-2016	References to GSMA IoT Security Assessment scheme added. Minor editorial corrections.	PSMC	Ian Smith GSMA
2.0	29-Sep-2017	Additional oneM2M references added	IoT Security Group	Rob Childs GSMA

A.2 Other Information

Type	Description
Document Owner	GSMA IoT Programme
Contact	Rob Childs – GSMA

It is our intention to provide a quality product for your use. If you find any errors or omissions, please contact us with your comments. You may notify us at prd@gsma.com

Your comments or suggestions & questions are always welcome.