An NCC Group Publication

Security of Things:

An Implementers' Guide to Cyber-Security for Internet of Things Devices and Beyond

Prepared by:

Ollie Whitehouse

Contents

1 Introduction

This white paper outlines a set of practical and pragmatic security considerations for organisations designing, developing and, testing Internet of Things (IoT) devices and solutions. The purpose of this white paper is to provide practical advice for consideration as part of the product development lifecycle.

While IoT products by their very nature encompass many forms of traditional embedded devices and supporting systems, we felt that distilling our knowledge and experience in the specific context of IoT would be useful. A lot of the concepts in this paper could easily be applied to many other related areas of software and hardware product development.

2 Defining the Internet of Things

Defined by CASAGRAS[1], the Internet of Things is understood to be "*a global network infrastructure, linking physical and virtual objects through the exploitation of data capture and communication capabilities. This infrastructure includes existing and evolving Internet and network developments. It will offer specific object-identification, sensor and connection capability as the basis for the development of independent cooperative services and applications. These will be characterised by a high degree of autonomous data capture, event transfer, network connectivity and interoperability*."

For the purposes of this white paper we define the Internet of Things as non-traditional personal computing devices connected to the Internet either directly or indirectly. This is a very broad definition, but is designed to encompass all embedded computing devices with Internet connectivity that may be interacted with by humans or machines. The methods of interaction may include machine-to-machine, sensor-to-machine, and user-to-machine. Internet of Things devices are typically not end-user serviceable. Examples of Internet of Things uses include but are not limited to:

- **Personal:** Health and activity monitoring and wearable computing.
- **Vehicular:** Telematics, control systems, and other supporting systems for road, rail, maritime, and air transport.
- **Smart Homes**: Home networks home automation, security, smart meters, entertainment, and domestic appliances.
- **Smart Buildings:** Electricity and building management systems. Electronic access management and monitoring.
- **Smart Cities**: Food safety, power generation and distribution, transit supporting systems and infrastructure, population (monitoring, management, and control), telecommunications, and emergency services supporting systems.
- **Smart Business:** Supply chain management, distribution, telepresence, and document management.
- **Health Care Life Critical**: Life supporting, monitoring, and diagnostics systems used within the health sector.
- **Industrial:** Robotics, industrial automation.
- **Space:** Satellites, space vehicles, and associated supporting systems.

How these devices communicate with the Internet will vary both in nature and frequency, from infrequent and short lived connectivity in the case of RFID, to the always-connected. However, all present opportunities both for end-users and those with malicious intent.

[1] Coordination and support action for global RFID-related activities and standardisation

3 Why Cyber Security Matters in the IoT

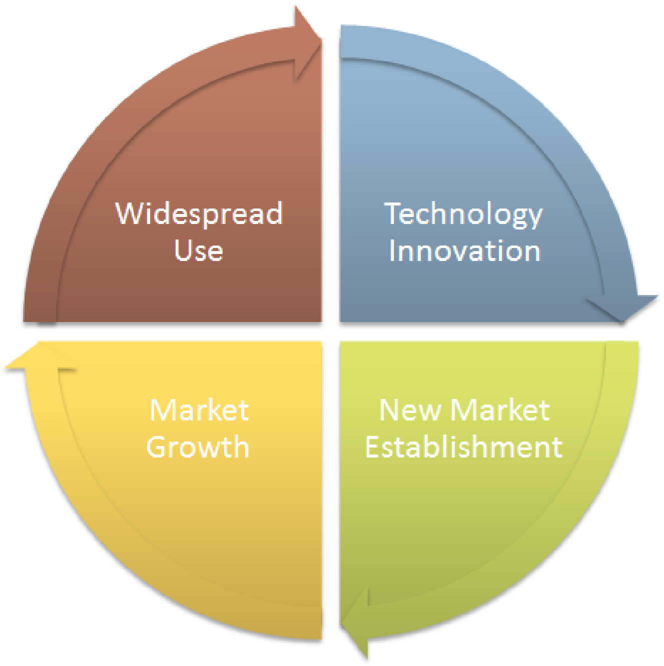
With the rise of the IoT, and with it the increased production and diversity of devices, economics and good business dictate that the development should be done as cheaply as possible. Unfortunately, if history has taught us one thing it is that even if working in regulated or highly cyber-security-sensitive markets, cyber-security will typically be dealt with in an uninformed manner. The regulators will often stipulate requirements that can be trivially met without achieving the desired or intended level of cyber-security (e.g. certain financial services regulations), or a buyer will look for documented security capability (e.g. encryption of a particular strength) rather than the quality of implementation.

What we have learnt over the past two or more decades is that:

- Security potentially carries with it a significant cost to product development of ~fourteen per cent[2] if done comprehensively. This figure is based on one previous study conducted by the author.
- Product managers, architects, developers, and testers rarely have deep cyber-security knowledge and skills.
- Time-to-market considerations will typically detract from product security.
- The lifetime of a product, if successful, will go far beyond that envisaged or desired by the vendor from a sustainment, maintenance and support perspective.
- Threat actor capability aggressively evolves, resulting in longer-lived systems becoming increasingly vulnerable with time, yet typically without the vendor sustainment support required to ensure ongoing security, integrity, and availability if targeted.
- If security is to be incorporated it will rarely be done from the start and often added at a later stage, reducing its effectiveness.

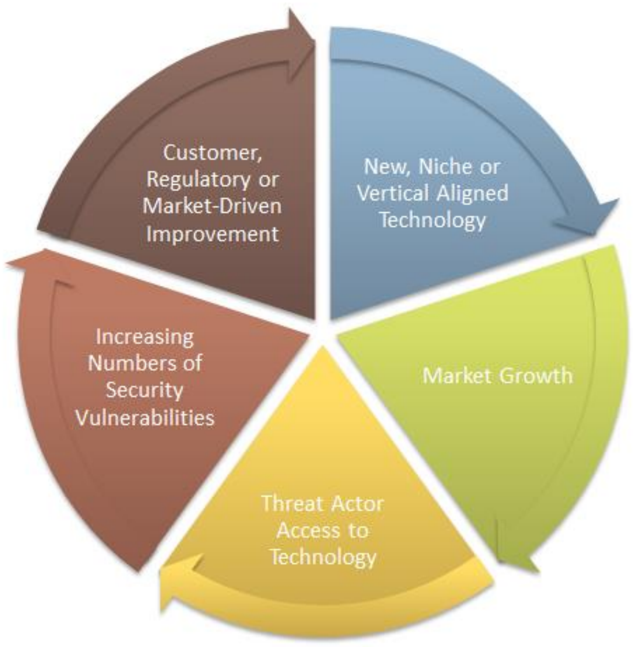
These lessons, combined with the fact a lot of IoT devices will be embedded within homes, buildings, urban environments, and business processes with a slow rate of replacement, means that those built over the next decade have a very real chance of being in use for ten or more years after release and in a lot of cases far longer. This situation leads to a dilemma, the answer to which is contrary to traditional wisdom regarding technical and security debt.

Historically product development follows the following evolution:



[2] http://recxltd.blogspot.co.uk/2012/01/cost-of-following-sdl.html

With traditional end-user software and frequently-replaced hardware this product evolution has allowed for an iterative approach to cyber-security. A product can be released with technical and security debt[3][4] [5] and then, as shown below, as threat actors gain exposure, learn about, and focus on the technology or product, allow just-in-time or post-event payback or shedding of the debt. This makes sense from an economics perspective as there is little upfront investment in an area with little or no perceived value. Money is only spent if the product becomes successful, thus maintaining market competitiveness.


New, Niche or Vertical Aligned Technology
Market Growth
Threat Actor Access to Technology
Increasing Numbers of Security Vulnerabilities
Customer, Regulatory or Market-Driven Improvement

The overarching problems with this approach were discussed in a post in 2011 titled *Breaking the Inevitable Niche/Vertical Technology Security Vulnerability Lifecycle*[6]. In the context of the IoT, however, cyber-security, while it should be considered a paramount requirement due to the pervasive and potentially long-lived nature of the devices, cannot and will not be regulated for in any meaningful way in the short term due to the concern of creating uncompetitive markets for product development and adoption plus the perception of risk.

So we are left with the following dilemma:

- Embed security throughout the lifecycle of IoT product development, resulting in higher costs and slower time to market yet clearly adding value in the short, medium, and long term.
- Develop products quickly to gain time-to-market advantage, with the markets and applicable regulators dictating requirements and thus the level of investment in product security by vendors.

Neither of which are particularly ideal. But suffice to say, unless cyber-security is considered in the requirements, design, developmental, and sustainment phases of IoT products, the problems we have faced with embedded systems from the last two decades will only increase.

[3] Security debt is security focused technical debt. The original paper outlined considerations and strategies for the management, payback and expiry of security debt as part of a mature security aware development lifecycle.
[4] http://www.amazon.co.uk/Software-Security-Austerity-development-ebook/dp/B007H76ABC/
[5] https://www.youtube.com/watch?v=W9DENKWHQVw
[6] http://recxltd.blogspot.co.uk/2011/12/breaking-inevitable-nichevertical.html

We do however believe that with rapidly evolving marketplaces, regulatory understanding, legislative, and legal systems the current approach of either building security in late or shipping now and patching later is not sustainable; thus for IoT implementers cyber-security should be considered from the outset, using appropriate risk-management and sustainment strategies.

Finally, if security is not addressed in the IoT, there is a very real risk that instead of benefiting from the gains in security we have seen in desktop and mobile computing we will instead regress, creating a larger attack surface of devices that are more vulnerable or provide ways into larger systems. This presents us with a very real risk that if security is not taken seriously by regulators, legislators, and markets then we will instead be in a worse situation than we are today. Security matters in the IoT products of today and tomorrow.

4 Why Privacy Matters in the IoT

As with cyber-security, we can see that privacy is also seeing legislative and industry body[7] interest in the mobile app sector. By extension, we can reasonably expect, and are indeed seeing, that the IoT will come under similar scrutiny[8]. Indeed the European Commission have published a fact sheet[9] on privacy and security in which they state:

It can reasonably be forecast, that if IoT is not designed from the start to meet suitable detailed requirements that underpin

- *the right of deletion*
- *the right to be forgotten*
- *data portability*
- *privacy and*
- *data protection principles*

then we will face the problem of misuse of IoT systems and consumer detriment

So for implementers of IoT devices and solutions it is important that privacy be considered. What this means in reality is ensuring that the areas listed above are understood, that only required data is captured, that personally identifiable information (PII) is collected and processed in a verifiable manner, and that PII is anonymised where possible.

By considering these areas proactively, implementers will be able to reduce the likelihood of future problems.

5 Secure by Default

Before delving into IoT specifics it is useful to be aware of the concept of “secure by default”. This term is described as[10]:

Security by default, in software, means that the default configuration settings are the most secure settings possible, which are not necessarily the most user friendly settings.

This definition highlights one of the key challenges: security versus usability. However, we have seen significant dividends in reducing the initial attack surfaces of products through the adoption of such strategies.

The UK Government department CESG provide specific advice and high-level requirements for the embedded hardware in devices in their May 2012 paper *Secure by Default: Platforms*[11] in which they state:

Platforms that are not secure by default cannot represent commercial good practice, and should not be relied upon to protect sensitive data.

[7] http://www.gsma.com/publicpolicy/privacy-design-guidelines-for-mobile-application-development
[8] http://www.iot-a.eu/public/news/internet-of-things-holds-promise-but-sparks-privacy-concerns
[9] http://ec.europa.eu/information_society/newsroom/cf/dae/document.cfm?doc_id=1753
[10] http://en.wikipedia.org/wiki/Secure_by_default
[11] https://www.cesg.gov.uk/publications/Documents/platforms_secure_by_default.pdf

… this paper will discuss desired characteristics of secure platforms; showing some ideas on how a concerted effort might be made to drive fundamental improvements in platform security. We aim to reduce the risk of a single vulnerability allowing a platform to be exploited. We will also suggest a practical example of how some of these characteristics could be provided.

The hybrid of these two concepts should be the aspirational goal of any IoT implementer factoring in the appropriate threats, desired functionality, impact of a security failure, and available resources.

6 Security Development Lifecycles versus Security Maturity Models

While not the focus of this paper, it is important to understand the difference between a Security Development Lifecycle, as originally developed and refined by Microsoft,[12] and a software security maturity model, as developed and refined in BSIMM[13].

This paper draws from the concepts of both of these approaches along with our own experiences, to provide a pragmatic set of recommended activities for implementers of IoT devices and solutions.

6.1 Security Development Lifecycle (SDLC)

A Security Development Lifecycle such as Microsoft's defines activities which should occur related at different stages in the development lifecycle. This approach is what is reflected in some of the activities detailed in this paper.

6.2 Maturity Models

A maturity model, such as the Building Security in Maturity Model, aims *to quantify the activities carried out by real software security initiatives*. This maturity model is built around five domains: governance, intelligence, SSDL (Secure Software Development Lifecycle – akin to an SDLC), touch points, and deployment.

7 Secure Hardware versus Open Hardware

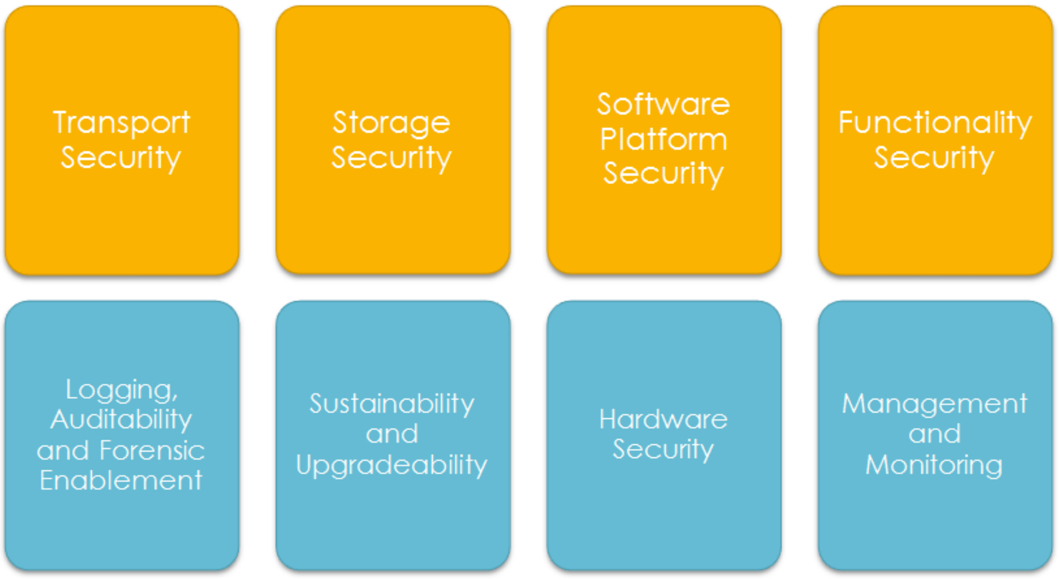
With the rise of the maker, hacker, and hobbyist hardware and software modification movements, and the desire for some vendors to facilitate innovation using their devices and solutions, the trade-off between security and openness is a common one. There is often thought to be a decision that needs to be made between security and openness. This does not, however, need to be an either/or decision. It is possible to design hardware products that by default will be open and allow software modification, while optionally providing end-user organisations the ability to lock the hardware down (e.g. via efuse use) to only accept signed firmware. Similarly, intelligence can be built into the boot ROM to make decisions on if certain regions of hardware storage or features are accessible if unsigned code is used. Alternatively, a trusted execution environment may be used to provide separation between the most sensitive operations and the general computing features (the general purpose OS) of the system or platform.

It is important that if a secure platform is needed in certain situations on devices which would otherwise allow such modifications, the requirement should be captured early on in the product's development lifecycle, as it will have numerous trickle-down implications through design and architecture, implementation, and testing.

[12] https://www.microsoft.com/security/sdl/default.aspx?mstLocPickShow=True
[13] http://bsimm.com/

8 Cyber Security Pillars for Internet of Things Products

When looking at the Internet of Things we can define a number of basic security controls which products should consider and implement proportionally to the sensitivity of the data they receive, hold, process, and transmit.


Transport Security
Storage Security
Software Platform Security
Functionality Security
Logging, Auditability and Forensic Enablement
Sustainability and Upgradeability
Hardware Security
Management and Monitoring

These pillars translate to:

- **Transport Security**: provide the appropriate level of identification, privacy, and integrity to network communication.
- **Storage Security**: provide the appropriate level of protection to persistent data held on the device or within the system.
- **Software Platform Security and Implementation**: select and implement platforms and supporting technologies that provide a robust and layered environment upon which to build the solution easily and quickly.
- **Functionality Security and Implementation**: implement functionality using a technology stack and tools which enable it to be done so in a secure fashion.
- **Logging, Auditability, and Forensics Enablement**: concrete sources of logs from low-level and high-level software components which facilitate investigation of misuse.
- **Sustainability and Upgradeability:** features which facilitate the ability to securely upgrade devices when vulnerabilities are discovered after release.
- **Hardware Platform Security:** ensuring the hardware platform provides the required security features.
- **Managing and Monitoring:** ensuring that IoT devices can be securely managed and monitored.

9 Cyber-Security Threat Landscape for the Internet of Things

Before deciding what level of investment is appropriate, it is important to understand the threats that Internet of Things products will face. The table that follows represents the most common threat classes that are likely to need considering. Many threats exist today but have a low risk of occurring; it is possible, however, that the threat landscape may evolve to make those threats more likely in the future.

Threat	Description	Impact
Compromise: remote	Compromise of the device and its data, either partially or entirely, typically over a network.	External security boundary is breached.
Compromise: local	Compromise of the device or its data, either partially or entirely locally, through either hardware or software means.	External security boundary is breached.
Privilege escalation	Increase in access, either locally or remotely, breaching a security boundary.	Degradation or failure of a security boundary leading to an increased level of access either on a temporary or permanent basis.
Impersonation	Impersonation of a trusted entity.	Degradation or failure of a security boundary leading to an increased level of access either on a temporary or permanent basis.
Persistence	Persistent access is obtained post-compromise through configuration modification or hardware / software manipulation.	Integrity of the platform or the external security boundary enforcement is no longer effective.
Denial of service	Service is lost, either partially or entirely, on a temporary or a permanent basis.	Degradation in availability or functionality.
Traffic interception or modification	Network traffic of any type can be intercepted, or modified.	Underlying trust in the integrity and privacy of the data traversing the network can no longer be guaranteed.
Stored data access or modification	Persistent data is read or modified.	Underlying trust in the integrity and privacy of the persisted data can no longer be guaranteed.

These eight classes represent in a distilled form the threats that IoT product designers and implementers will have to contend with. As is clear from the above, any one of these can be serious depending on the function of the intended device. The majority of security issues that NCC Group identifies can point back to one of these threat classes. Microsoft's perspective on the same problem uses the acronym STRIDE[14], which implementers may alternatively consider.

As previously stated, these threat families do not capture the risk that the events may occur; however it is our opinion that all should be considered and appropriate risk analysis be performed.

10 IoT Devices and System Integration Security

While much attention has been paid to the rise in the devices that become the things of the Internet, less is paid to the systems with which they integrate. So while this paper talks about many of the threats in the device context, care and attention should be given to the integrated system as a whole. This is especially true where aggregation and processing of data occurs, as it is this concentration of data or functionality that represents the biggest risk; for example the device or its access being used as an entry or pivot point with which to attack the larger system.

The consequences and risks to these larger systems should not be downplayed or dismissed. In NCC Group's experience these systems are the ones which have wide reaching consequences if compromised and are also often built using a mixture of legacy and new technology and systems. This evolving nature of systems leads to a perceived fragility and criticality which often precludes thorough analysis and attack simulation resulting in reliance on high-level risk management approaches. As a result these paper based exercises in risk management and mitigation are relied upon which do little to mitigate or address the real-world technical risks and attack scenarios.

Understanding where and how the IoT will integrate with such systems, their level of access and influence, and the ongoing data paths and relationships are critical in gaining a picture of the true exposure.

11 Other Considerations

There are a number of other considerations with regards to device functionality, use cases and potential impacts which, while not the focus of this white paper, will likely have a bearing on any threat modelling undertaken.

11.1 Public and Personal Safety and Security

It's increasingly likely that with the advent of autonomous machine-to-machine devices that more and more will be related to safety and security. The uses of these devices will range from critical health monitoring sensors (life critical), in-car networks, and associated safety devices through to electronic locks and environment security monitoring. It is imperative to understand the potential for physical or environment harm or loss of life from misuse, electronic compromise, or denial of service, to facilitate accurate threat modelling.

11.2 Ad Hoc and Transient Relationships

Understanding machine-to-machine and user-to-machine use cases as well as security and privacy and the *ad hoc* and transient relationships that may exist will likely bring to the surface a number of considerations. These *ad hoc* and transient relationships may mean implicit trust does not exist or that the relationship may be short lived resulting in the need to adjust behaviours or assumptions. Examples of *ad hoc* and transient relationships may include:

- A mobile phone temporarily linked to a rental car's hands-free kit.
- A sensor network built for a sporting event or emergency service's major incident response.
- Personal health monitoring network temporarily connecting to a public Wi-Fi hot spot.

[14] http://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx

- NFC tag making contact with an NFC reader in a public environment.
- ZigBee *ad hoc* networking used to construct a gym network of health monitoring devices.

These scenarios would all require subtle changes in approach to ensure security and privacy is proportionate to the use case and in line with user expectations.

11.3 Underlying Transport Infrastructure Security and Resiliency

It's also important to understand the assumptions you will be making around the underlying transport infrastructure security and resiliency. If your product will have a lifespan of many years, either intentionally or not, assumptions made today can potentially cause significant problems in the future. A case in point is with UDP services on network devices that provide a degree of amplification; that is to say they respond with more data than they are supplied with when communicated with over UDP. The problem arises in this case because UDP is connectionless and thus the source address can be spoofed. Amplification, combined with this ability to spoof the source address from which requests are made, provides a threat actor with a powerful denial of service capability. Thus any device which implements such a service potentially contributes to the instability of the Internet. Another example would be where assumptions are made that transport security will be guaranteed forever or at least the lifetime of the product. As we have seen with GSM, WEP and a number of other wireless protocols this assumption would have been proven incorrect.

11.4 Internet Environment Footprint and Impact

When designing a device and its supporting services, it is important as the vendor that you understand what the footprint and the potential impact on the Internet will be if successful, widely deployed, or heavily used. Understanding how the device and service will behave in bandwidth-constrained or higher-latency environments and the potential impact if it malfunctions are all activities designed to ensure that your device or service does not negatively impact the wider Internet.

11.5 Intellectual Property Protection

Devices may be built with highly-valuable intellectual property that the vendor wishes to protect, or alternatively may contain and process intellectual property that the rights owner want protected. There are two important things to consider here. The first is that it is near impossible to fully protect intellectual property when the device is in the physical possession of a threat actor. Most defensive strategies involve slowing down or otherwise degrading a capability from a break-once-use-everywhere to break-once-use-once. Secondly, effective technical defences for intellectual property protection will rely on solid security foundations, much of which are outlined in this paper.

12 Practical Threat Modelling and Risk Assessments for Internet of Things Product Development

In order to perform a practical risk assessment, and thus understand where investment should be made, it is important to take a number of distinct areas into consideration.

12.1 Risk Areas

Risk areas can be thought of as those elements of a device or system that influence the level of risk.

Risk Area	Questions to Answer	Example
Accessibility	How accessible is the device in terms of functionality and network connectivity? How accessible is the device from the network? How accessible is the device in terms of end-user interaction?	An RFID tag will have a low degree of accessibility, requiring very close proximity to the tag. A road traffic speed monitor sensor will have a low degree of accessibility due to a sole pressure plate input and a single outbound network connection to stream the data. A SmartMeter would have a high degree of accessibility in terms of both user-accessible features and range of network services and functions.
Integrity	How integral do the device, the data the device holds and processes, and any network traffic need to be? What are the implications on wider decision-making systems if varying amounts of corrupt or deliberately malformed data were supplied?	A non-life-critical sensor used to perform high-resolution statistics collection will be likely to have a low integrity requirement due to the number of measurement points. A smart meter which is used for monitoring, billing, and remote control will have high integrity requirements due to the ramifications if network traffic is manipulated
Identity and non-repudiation	To what degree do identity and non-repudiation need to exist within the system? What are the implications if traffic or data can be spoofed on behalf of another device?	A temperature monitor in a home will likely carry a low requirement of identity and non-repudiation. A car tyre monitor which will trigger limp mode in a vehicle will have a requirement for a medium degree of identity and non-repudiation. A payment RFID tag will have a requirement for a high degree of identity and non-repudiation.

Risk Area	Questions to Answer	Example
Confidentiality	How confidential is the data that the device stores, transmits, or receives? What is the privacy implication from a single instance or aggregation if the data produced is compromised?	A home-based sprinkler system would have low confidentiality requirements. An in-car map system which co-ordinates with the user's diary, live traffic patterns, and historic routes driven would have a high confidentiality requirement.
Availability	How critical is it that the device be available and functioning? What are the implications from a single instance becoming unavailable? Do those implications change for multiple instances?	A roadside pollution sensor would be likely to have a low availability requirement. A critical care health monitor would have a high availability requirement.
Environment	How secure is the environment in which the device will be operating? What are the implications on the security of the instances or wider system should one be comprehensively reverse-engineered?	A personnel movement tracker within an office building to aid smart power and environmental control would be operating in a secure environment. A metro transport RFID tag that is responsible for payment and issued to the general public would be operating in an insecure environment.
Safety	Is there a risk that humans or animals could be harmed through electronic control?	A car with a CAN bus network that is connected to the airbags and telematics systems which in turn is connected to the Internet could put lives at risk. Should the telematics system be compromised via the Internet then the airbags could be remotely deployed injuring the driver and putting the passengers or other road users at risk.

After the risk areas have been identified and relevant questions answered, they will act as a backbone to a number of security decisions, modelling exercises, and analyses.

12.2 Threat Actors

Before performing the risk analysis, another important factor to consider is the threat actors, their motivations and capabilities. While this is ever-changing, it is important to acknowledge the nature of the likely threat actors, as there will be a significant amount of time, effort, and money required to defend against them.

Threat Actor	Motivations	High-Level Capabilities	Cost of Defence
High school student	Intellectual challenge and mischief	Large time budgets, Internet sources, programming, software reverse engineering, basic hardware reverse engineering, very limited budgets and peer support.	Low
Curious consumer	General interest	Internet sources, programming, software reverse engineering, basic hardware reverse engineering, very limited budgets and peer support.	Low
University student	Intellectual challenge, studies, and peer recognition	Large time budgets, Internet sources, programming, software reverse engineering, advanced hardware reverse engineering, high-performance computing, limited budgets and peer support.	Low
PhD researcher	Intellectual challenge, studies, peer recognition, industry recognition, media coverage, and commercial funding providing directed goals	Internet sources, programming, software reverse engineering, advanced hardware reverse engineering, high-performance computing, variable budgets, experience, and peer support.	Moderate
Commercial researcher	Intellectual challenge, peer recognition, industry recognition, media coverage, and commercial goals based on commercial or government client requirements.	Internet sources, programming, software reverse engineering, advanced hardware reverse engineering, high-performance computing, experience, variable budgets and peer support.	Moderate
Political activist	Investigation, research, and propaganda / influence	Internet sources, programming, software reverse engineering, advanced hardware reverse engineering, high-performance computing, experience, variable budgets, peer support, and sympathetic insiders.	Moderate

Threat Actor	Motivations	High-Level Capabilities	Cost of Defence
Organised criminal	Intelligence and monetisation	Internet sources, programming, software reverse engineering, advanced hardware reverse engineering, high-performance computing, experience, significant budgets, peer support, organisation, ability to plant individuals within organisations allowing access to internal information, and enhanced technical capabilities.	Moderate to high
Low-grade terrorism	Criminal enterprises, counter intelligence, propaganda / influence, disruption, and fear	Internet sources, programming, software reverse engineering, hardware reverse engineering, high-performance computing, experience, significant budgets, and peer and sympathiser support.	Moderate to high
High-grade terrorism		Internet sources, programming, software reverse engineering, advanced hardware reverse engineering, high-performance computing, experience, significant budgets, peer support, organisation, ability to plant individuals within organisations allowing access to internal information and enhanced technical capabilities.	High
Commercial competitor	Intellectual property and competitive analysis	Internet sources, programming, software reverse engineering, advanced hardware reverse engineering, high-performance computing, experience, significant budgets, peer support, organisation and enhanced technical capabilities.	High
Low-grade nation state	Intelligence, counter-intelligence, propaganda / influence, military capability, and economic advantage	Internet sources, programming, software reverse engineering, hardware reverse engineering, high-performance computing, experience, significant budgets, and peer and sympathiser support.	High

Threat Actor	Motivations	High-Level Capabilities	Cost of Defence
High-grade nation state		Internet sources, programming, software reverse engineering, advanced hardware reverse engineering, high-performance computing, experience, significant budgets, peer support, organisation, ability to plant individuals within organisations allowing access to internal information, ability to influence supply chain, and highly advanced technical capabilities, including development of new attack techniques.	Very high

12.3 Attack Trees and Asset-Centric High-Level Threat Models

Another two important points to consider during the early phases of product development are attack trees and asset-centric high-level threat models. While impossible to create entirely early on, they will evolve over time as more becomes known and decisions are made. Starting to construct these attack trees earlier rather than later can help inform decisions and stimulate debate whilst showing possible attacks and how they could occur.

12.4 Practical First-Pass Risk Analysis

Traditional risk analysis uses a variety of different models, factoring in a number of different elements such as impact and likelihood. In order to determine the risk, there may also be secondary inputs such as the value of the asset and the nature of the vulnerability (if known).

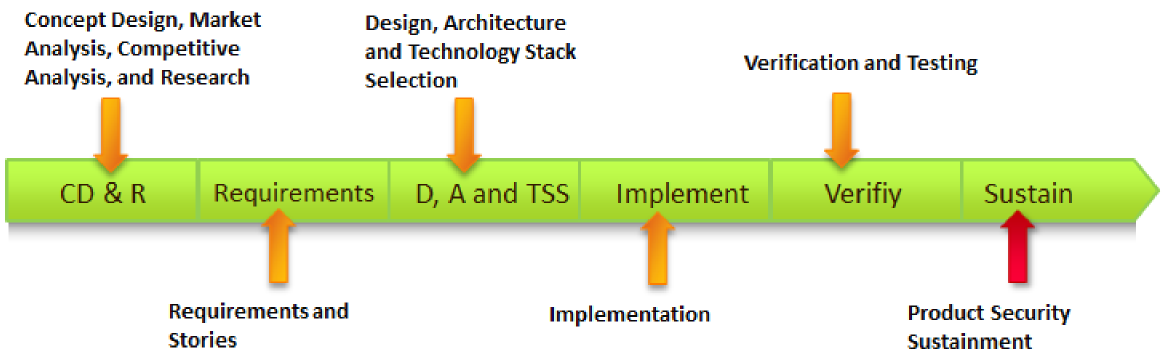
In our experience, however, these exercises rarely represent the real-world risk, due to lack of understanding of both the impact and the likelihood by those conducting the exercises. Instead we would suggest that a practical risk analysis for a new IoT product should consider:

- Threat actors the vendor wishes to reasonably protect against.
- Risk area considerations.
- Threat landscape elements which apply.
- Attack tree and asset-centric high-level threat models.

All of these areas will then influence the requirements, design and architecture, implementation, testing, and sustainment phases of the project. Our experience shows that a simple risk equation/model approach at the very start of a project is not suitable for complex devices and systems. Instead, a number of discrete discussions need to happen with various stake holders. Once the common security aims, risk areas, and threat actors against which defence is necessary have been agreed, further detailed risk analysis and threat modelling can occur in the later phases.

13 Product Lifecycle Phases/Sprints and Cyber-Security

In the following section we outline for implementers the types of cyber-security-supporting decisions and activities that it is recommended should occur during the different product lifecycle phases. The purpose of this is to provide practical advice and guidance to help ensure cyber-security is both presented and considered throughout the development of the product, while also providing technical considerations for implementers.


Concept Design, Market Analysis, Competitive Analysis, and Research
Design, Architecture and Technology Stack Selection
Verification and Testing
CD & R
Requirements
D, A and TSS
Implement
Verifiy
Sustain
Requirements and Stories
Implementation
Product Security Sustainment

13.1 Phase 1: Concept Design, Market Analysis, Competitive Analysis, and Research

During the initial phase of product development there may be some amount of concept design, market analysis, competitive analysis, and supporting research. During this phase the role of a security-aware individual is normally light touch, providing consultancy, analysis and advice around the high-level concepts, components, and technology security capabilities. Security advice may also be taken regarding competitor capabilities and the expected direction of market, regulatory, and legislative forces with regards to cyber-security.

Even at this early stage this insight can be invaluable, as it will provide very high-level inputs that will help determine if the proposed concepts are viable and what the considerations should be during the requirements phase.

13.1.1 Cyber-Security Related Actions and Activities

- Provide pertinent market, regulatory and legislative cyber-security insight and research.
- Assess competitor cyber-security capability and market differentiators.

13.2 Phase 2: Requirements and Stories

The requirements phase for an IoT or other embedded product will typically involve security in two ways. Firstly, usability, physical form factor, power consumption, and other technological requirements are likely to place constraints on the ability to secure the platform. Secondly, individuals with a security mandate should be part of the team participating in the formulation of the technical and market requirements. It is important that those with security responsibilities examine all requirements to help highlight those that are diametrically opposed to the security and privacy goals of the product or carry with them a degree of risk or exposure.

It is during this phase that a number of key decisions will be made with regards to the trade-off of usability, deployability, manageability, compatibility, and security. These trade-offs are to be expected, but the risks should be formally documented and reasons given for their acceptance.

For example, in the case of an IoT device the type of RFID tag required or wireless protocol such as Bluetooth LE may be known to be vulnerable or otherwise have a weak security posture; however, due to compatibility or cost considerations, this is a known and accepted risk.

13.2.1 Cyber-Security Related Actions and Activities

- Participate in requirements discussions to provide a balanced cyber-security perspective.

- Provide high-level market and technical cyber-security requirements and stories.
- Review other requirements to identify potential security risks and exposures, understanding they may be acknowledged and accepted and the risk born due to overriding factors.

14 Phase 3: Design, Architecture and Technology Stack Selection

14.1.1 Hardware

When translating the requirements and stories into design, architecture, and technology stack selection activities for hardware, the considerations listed below are recommended to be taken into account proportionate to the risk. As previously stated, such consideration should be proportionate, as most will add cost due to development and bill of materials, form-factor restrictions, and, if the device is battery-powered, potential impact on longevity or increased battery requirements.

It is important to keep in mind that with hardware any attempt at security via obscurity or masking certain interfaces will almost be guaranteed to be discovered if the device is targeted in any meaningful manner.

Hardware Component	Consideration	Reasoning
Boot ROM	Trusted or verified boot	For devices that do require a trusted or verified boot it is important that the trust anchor be as low as possible. In some processors this boot ROM will be inside the CPU itself while others may rely on an external boot ROM.
CPU/DSP/Microcontroller/SoC/RF SoC	Hardware-accelerated cryptography	For devices requiring heavy use of cryptography, hardware implementations can improve performance and extend battery life. However, it is important to understand which ciphers are supported and if they provide a reasonable lifetime for the expected length of the product.

Hardware Component	Consideration	Reasoning
	Hardware PRNG and entropy	Most devices will use random numbers extensively. If these need to be strong, then the available entropy from software-only sources can be low upon first startup. It is important to understand what, if any, hardware PRNG or entropy sources are present and their quality.
	Trusted or verified boot	For devices that do require a trusted or verified boot it is important that the trust anchor be as low as possible. In the case of some processors this will be inside the CPU itself while others may rely on an external boot ROM.
	Privilege levels, rings or domains	Ability to separate kernel and userland with virtual memory addressing and similar within userland. Not common on microcontrollers but essential for defence in depth.
	Exploit mitigations	Features such as overflow flags, non-executable memory, and similar are required to enable software mitigations.

Hardware Component	Consideration	Reasoning
	PIN out placement	For chips with security features or functionality that may impact security it is important to understand where these are located on the chip's pin out. It is generally advisable not to use chips where these features are on the outer two rows in high-security environments due to risk of fly wires being used.
	Efuses and similar	Efuses and similar features can provide vendors with the ability to disable certain hardware features, burn in identities, or use non-reversible sources of configuration data for software. Understanding these properties of the processors being used is imperative.
	Trusted execution environment	For high-security applications it may be desirable to have a trusted execution
Memory controller e.g. MMU or ioMMU	Trusted execution environment support	Where a trusted execution environment will be used it is mandatory in most cases that the memory controller also supports it.
	DMA/IO control	Where DMA will be used it is important from a security perspective to be able to restrict which regions are addressable from certain peripherals. This is to ensure that sensitive memory cannot be read or written unless required.

Hardware Component	Consideration	Reasoning
Circuit design	Via exposure	Vias provide an attack service to the circuit board. As a result, their exposure should be considered and possible attacks mitigated. Where possible, blind and buried vias should be used.
	Bus lines	All bus lines should be considered to determine what data they will expose and potentially allow to be read or modified. Those deemed of a highly security-impacting nature should normally be buried.
Interfaces	JTAG interface security	JTAG provides low-level access to components and can often be used to subvert various security features. This exposure needs to be understood and where appropriate mitigated.
	I2C/SPI bus monitoring and manipulation	I2C/SPI buses are often used for inter-peripheral communication. Often they are easy to sniff and interact with, resulting in a range or impacts. This exposure needs to be understood and where appropriate mitigated.
	Serial interface security	Serial interfaces are often used to provide some form of console access for debugging, development, and support. They also provide a common attack vector and thus need to be considered carefully.

Hardware Component	Consideration	Reasoning
	BootROM interface security	Any interfaces, be they console or logical, over USB or serial, need to be considered in the context of security. Boot ROMs often provide a way to load firmware or otherwise manipulate the boot process, both of which potentially present security issues.
	Firmware update interface security	Interfaces which provide the ability to flash, update, or otherwise modify firmware, either running or persistently stored, need to be considered due to risk of modification or misuse.
	Configuration and calibration interfaces	Any hardware interfaces used for factory or aftermarket configuration and calibration need to have their security impacts understood.
	Inter-processor IPC	Hardware interfaces which provide inter-processor IPC mechanisms similar to generic bus lines need to be carefully reviewed. IPC between processors is often implicitly trusted and provides an often-fertile attack surface.
Persistent storage	Secure storage for data at rest	Where data needs to be secure in the case of physical attack due to loss or otherwise, understanding how sensitive data will be secured, both from an integrity and privacy perspective, is important.

Hardware Component	Consideration	Reasoning
	Performance	When using code signing and other secure features, persistent storage performance can often become a bottleneck where many reads are required. As a result careful consideration around performance should be given.
	Secure erase and wear levelling	Where secure destruction of persisted data is required, it is important to understand wear-levelling behaviour and the ability to overwrite wear-levelled blocks in a secure fashion.
Component packaging	Ease of access to memory, flash, and CPU	Selecting certain system-on-chip packaging options where, for example, memory is physically located under the CPU, can complicate hardware attacks that would otherwise be trivial.
Anti-tamper / tamper detection	Suitability for implementation and, if suitable, the degree to which they should be implemented. Such defences on the low end can be epoxy and similar materials. On the high-end there are meshed containers which detect intrusions and CPUs which used encrypted instructions.	High-security applications are likely to need to withstand more advanced reverse-engineering and attack methods. Where this is the case anti-tamper and tamper detection can provide hardware and software based defence in depth.

Hardware Component	Consideration	Reasoning
Wireless/RF	Security posture of wireless/RF standards which are to be used.	If inherent risks are identified in the wireless or RF technologies they can potentially be addressed and verified as being fixed during software feature design, implementation, and testing.
CPU/DSP/microcontroller/memory/persistent storage and buses	Side channel attack susceptibility including power analysis susceptibility and shielding	Side channel attacks such as external power analysis using a variety of techniques can yield sensitive information from a running system. For high-security applications a low end defence that could be considered is shielding. Whilst at the high end hardware and software mitigations should be put in place.
	Non-persistent storage behaviour and data storage degradation rates upon power source removal.	Certain attacks require the removal and transplantation of components from a target to an analysis system. Understanding the behaviour of the components with regards to non-persistent storage behaviour especially the rate at which data remains recoverable is an important security consideration.
	Microprobing susceptibility and mitigations	Microprobing can be used to access chip surfaces directly. This type of analysis can be used to observe, manipulate, and interfere with the targeted component.

Hardware Component	Consideration	Reasoning
Power management	Misuse to slow down systems to make glitching more reliable.	There have previously been examples where power management functionality which influenced CPU stepping has been misused to allow for hardware glitches to be made more reliable. Understanding these possibilities and if there is need to defend can be important in high security applications.
External packaging/casings/tamper evidence indicators	Tamper resistance and visual evidence	Designing external casings to either be resistant to physical attack or constructed in such a way as to require physical breakage to access a system's internals can limit attacks which only have short-lived access. Alternatively other tamper indicators may be useful, such as those that change colour on exposure to oxygen, moisture or heat. These indicators are designed to warn that tampering may have occurred.
All	Glitching attacks and similar	A common attack on exposed buses and components is to attempt to glitch (inject or corrupt) data or instructions in transit or execution. The ability to detect and defend such attacks should be considered for high-security environments.

Hardware Component	Consideration	Reasoning
All	Removability of components	The ease with which components can be removed will be a concern in high-security applications. Removing components may facilitate reverse engineering

While not an IoT device, we can see the ramifications of unintentional hardware interplay if we look at the reset glitch hack on the Microsoft XBOX 360[15]. In this case a reset line was pulsed to bypass a code-signing check and thus achieve unauthorised code execution using a hardware glitch. It was possible to make this exploit reliable by slowing down the CPU by a factor of 128 by injecting a specific signal. Alternatively, for an example of a hardware attack from 2009 that affected IoT devices we just need to look at Travis Goodspeed's work in the paper *Extracting Keys from Second Generation Zigbee Chips*[16]. In this paper Goodspeed demonstrated a local attack that allowed key theft from unprotected data memory. Both of these examples serve as a reminder of the possible ramifications of hardware attacks given a motivated threat actor.

14.1.2 Software

Depending on the type of IoT device and supporting solution being developed it is likely that there will be a variety of fundamental decisions to be made around the technology stack. Decisions made here can be critical in ensuring product security.

Component	Consideration	Reasoning
Programming language selection	Unmanaged languages, while faster, introduce the overhead of having to manage certain aspects that lead to classes of vulnerabilities such as overflows, use after free, dangling pointers, and similar.	Understanding the security considerations for the language can ensure they are accommodated in architecture, development, and testing.
Developer tooling	Developer tooling should facilitate secure coding, implementation of defensive techniques and leveraging of operating system defences. This is especially true when using unmanaged languages such as C and C++. This will involve using modern compilers with security options turned on, and IDEs and CI systems that can perform static code analysis.	Code that is identified as being insecure at the time of development is quicker and easier at point of creation. Modern compilers provide a defence-in-depth capability to unmanaged programming languages during the compilation process.

[15] http://libxenon.org/index.php?topic=155.0
[16] http://www.blackhat.com/presentations/bh-usa-09/GOODSPEED/BHUSA09-Goodspeed-ZigbeeChips-PAPER.pdf

Component	Consideration	Reasoning
Development frameworks	Ensure the frameworks selected enhance security rather than detract. These can include web frameworks that will reduce common vulnerability classes or native language frameworks that address common memory corruption vulnerability classes.	Development framework selection can either greatly reduce or increase the amount of security engineering and awareness required. depending on the situation.
Operating system or platform	Select a modern operating system or platform that provides defence-in-depth properties, including but not limited to ASLR, non-executable memory, process segregation, and sandboxing.	If secure coding practices and subsequent compiler protections fail to mitigate exploitation of a vulnerability, a modern operating system or platform that provides defence in depth can help minimise the impact and facilitate recovery to a known good state.
Development and debug platform and software interfaces	Ensure that development and debug interfaces cannot be misused in shipping products to subvert security.	Development and debugging interfaces typically provide low levels of access or sensitive information. As a result if they are included in the shipped product they may allow an attacker to subvert security measures.
Third-party component and library usage	Plan on how updates to third-party libraries will be tracked and integrated on an ongoing basis as security vulnerabilities are discovered.	Third-party library code is often a source of known vulnerabilities. Understanding how this information will be monitored and actioned is critical to a product security sustainment strategy.
Leveraging compiler, operating system, and platform security features	Plan on how to leverage all possible security features of the compiler, operating system, and platform. This will involve, in the case of complex operating systems, significant documentation review and planning due the potential impact of software architecture.	Designed to ensure all defence-in-depth features available are used.

To understand the ramifications of getting the above process wrong we only need to look at the example of the *PRNG Vulnerability of Z-Stack ZigBee SEP ECC*[17] discovered by Travis Goodspeed again in 2009. He concluded:

[17] http://travisgoodspeed.blogspot.co.uk/2009/12/prng-vulnerability-of-z-stack-zigbee.html

This article has shown that the Chipcon ZStack library, as of version 2.2.2-1.30 for CC2530, uses an insufficiently random PRNG for cryptographic signatures and session keys. PRNG data repeat every 32,767 samples, and there are at most 16 bits of entropy in any key. Searching the entire key space ought to be possible in very little time. Contrary to the CC2530's documentation, these random numbers must not be used to generate random keys used for security.

All users of this library, particularly those using it for Smart Grid devices and other industrial applications, are recommended to re-implement macMcuRandomWord() and to ensure that nothing requiring cryptographic security operates from the PRNG alone. Users of other libraries for the Chipcon devices should ensure that those libraries are not using the PRNG data.

This outcome is likely undesirable for any vendor, and especially those working in smart grid, and thus again serves as a good example of why such fundamental functionality should be verified during testing.

14.1.3 Functional Requirement Design and Architecture

The table below contains a list of common functional requirements which require design and architecture consideration. The table presents the security considerations associated with each and the reason for doing so. The purpose is to ensure that the right considerations are given to ensure an appropriate level of security.

Functional Requirement	Consideration	Reasoning
Installation and customisation	How the provisioning processing will happen in a secure fashion.	Insecure provisioning processes can potentially open up devices or systems to attack upon initiation.
Connectivity	Wireless and RF specification intrinsic security or lack thereof. Connectivity authentication. How will the connectivity be authenticated to? How will credentials be stored? Can the credentials be easily transplanted to another device?	If inherent risks are identified they can potentially be addressed and verified at higher-level communication design, implementation, and testing. Understanding how wireless authentication will occur, the credentials stored and if they can be transplanted are obviously very important. For example if cellular services are used and a SIM module is present it might be decided that a surface mount SIM module would be more secure than a removable one to mitigate easy movement between devices.
Communication	How communication will occur in line with the desired privacy and integrity requirements. Also how this communication will mitigate man-in-middle and similar attacks.	Assumptions around communication security can undermine the security of a platform; these are often difficult to fix retroactively, due to typical requirements to maintain compatibility.

Functional Requirement	Consideration	Reasoning
	Are the wireless, RF, and wired transports considered untrustworthy or likely to be undermined in the lifetime of the product, thus requiring higher-level mitigations?	Providing defence in depth against underlying protocol breaks is costly and complicated. If there is a high degree of trust in the lower level transports, then this can simplify and expedite development.
Encryption	What are the encryption requirements for storage and transport?	Understanding the requirements ensures appropriate selection of ciphers, modes, and key schemes.
	What are the hashing requirements for the products?	Understanding the requirements ensures appropriate selection of ciphers.
	Where a pseudo-random number generator is required what sources are available and what is their quality?	Weak pseudo-random number generators have been the root cause of a number of product security vulnerabilities over recent years.
	How will encryption keys be generated, stored and transmitted (as required)? What types of keys will be used (e.g. symmetric versus asymmetric)?	Encryption key type, generation, storage, transmission, and replacement are often done in an insecure manner, allowing product security breaks to occur.
	What are the available ciphers, and what is the quality of their implementation in both hardware and software?	Ensuring correct and robust ciphers are available to match the expected use case and strength requirements. Also how the ciphers will be used depending on the use case to ensure the expected level of security.
	Performance overhead and battery (if applicable) impact.	Ensuring the performance will not be negatively affected when using the required ciphers is typically an important consideration.

Functional Requirement	Consideration	Reasoning
Integrity	What are the integrity requirements for data at rest and in transit?	Understanding the integrity requirements for different aspects of the systems will influence the design and cost of the product. This understanding can influence software and hardware selection significantly.
Identification	How robust does the ability to identify the device and user need to be against cloning and similar attacks?	Relying on changeable values within software to uniquely identify a device greatly reduces the cost of cloning, and may facilitate hardware replacement or similar, thus undermining assumptions made in the design.
Non-repudiation	Do transactions or requests from the device or user need to be non-repudiable?	If transactions such as payment, purchases, or management requests need to be non-repudiable in nature, then considerations around cryptographic signatures or other non-repudiation mechanisms need to be taken into account.
Data destruction	How robust does data destruction on the device need to be, either as part of standard operation or in the case of compromise or loss?	Understanding the data destruction requirements will influence hardware component selection as well as the software implementation and testing phases.
Services	What network services will be exposed? What level of authentication will be required to access these services? What data or functionality will be exposed by these services? Do these services require an authorisation model as well as authentication?	The network-exposed services will present one of the largest attack surfaces of the product. Understanding what will be exposed, how, and any intended security will allow for reduction in this surface. It will also feed into threat-modelling exercises while heavily influencing implementation and testing.

Functional Requirement	Consideration	Reasoning
Service interaction	With which services will the device be interacting? Will this device be the weak link into the system by having elevated access? Does the device interact with those services in a secure fashion? Does the device need to verify the identity of a service before sending or receiving sensitive information? Does the device consume data from services from which untrusted or potentially malicious content may originate?	Taking into account all of these considerations will help develop the threat model while also highlighting areas and threats for implementation and testing to be aware of. Consuming data from services from which untrusted or potentially malicious content may originate may provide a viable attack surface, as we have seen in the desktop arena. Not verifying service identity before sending or receiving sensitive information allows for the potential of man-in-the-middle attacks. Devices which are assumed to be trusted or uncompromised may have elevated or otherwise enhanced access to an environment or system. If this is the case then careful consideration needs to be given to the ramifications if one or more devices were compromised.
Management	How, if at all, will the device be remotely managed? How will this management be done in a secure fashion?	Building on the service interaction considerations, device management should be considered a security risk.
Vendor Support	How will vendor support be provided?	Backdoors for vendor support have been discovered in a number of products. The existence of these backdoors has been discovered through reverse engineering. A vendor's support mechanism should be advertised, secure, and optionally be disabled by the user to enhance security.

Functional Requirement	Consideration	Reasoning
Upgrading	How will the product be upgraded in a secure and scalable fashion should future security vulnerabilities or other bugs require a software fix?	Software update mechanisms are often implemented insecurely or not present at all. Either of these would be detrimental to product security.
Diagnostics	How will diagnostics be performed on the device either locally or remotely in a secure fashion?	Diagnostic interfaces can be implemented in an insecure fashion, providing facilities or aids to compromise. By considering these factors during the design, secure diagnostics can be designed.
Logging and auditing	Should there be a need to forensically analyse a device following a compromise or for other reasons, how robust is the logging and what time period will it cover? How does a consumer or end-user organisation responsible for the devices know that they're under attack or have been attacked?	Increasingly IoT devices are being compromised and used as pivot points or platforms from which to launch attacks. Being able to investigate these in an effective manner is critical, but this requires the support of the vendors to provide robust logging and auditing information.
Backup and restore	What backup and restore functionality is required? What if any security impact will this functionality have? What authentication or authorisation will be required? Does transport security need to be considered?	Backup and restore functionality can provide a vector for obtaining sensitive information but also potentially compromising a device. Careful design should occur to ensure these risks are minimised.
Recoverability	If a device is compromised how can it be put into a known good and trustworthy state?	The historic wisdom about reinstalling a system following a compromise is not always practical or possible for IoT devices. So how will the design cater for putting it back into a known good state post-compromise?

14.1.4 Integration Security

When integrating IoT devices into a larger system, specific focus should be given as to how this will be done in a secure fashion. As with any system, understanding and defining trust boundaries is imperative. These trust boundaries will define where there is no, partial, or implicit trust on components used within the larger system. It is also common for the security of points of integration to be assumed to be the responsibility of the other parties, leading to gaps in responsibility, due diligence, or analysis.

As an example, we have seen a number of telecommunication operators fall foul of this specific problem. They provided access to key parts of their networks to femtocells distributed to users in their homes. However, as different threat actors worked out how to attack and compromise these femtocells via hardware and software means, this legitimate access that was extended to the device could be misused.

Another example is a simplistic sensor which simply transmits pollution data. It transmits this data, which is not particularly sensitive and does not have any impact other on than the quality of monitoring if spoofed, over the Internet. In this situation there will likely be little concern about a sensor being compromised, as it will have no elevated access within the system it is integrated with. As a result the security considerations would be far less.

It is important during the design and architecture stage to understand as part of the planned integration what if any elevated access or implicit trust will be placed on the integrity of the device this larger system communicates with. The ability to detect malicious activity from these devices and respond is also extremely important. Responses may include being able to de-provision or remove access to a particular device without impacting others.

14.1.5 Detailed Threat Modelling

Much has been written, including entire books[18], about the virtues of threat modelling early in the design stage of a product's development lifecycle. How to conduct such threat modelling is not the subject of this paper, but suffice to say that threat modelling should be undertaken at various levels, including system, device, and functional, as early as possible.

These threat-modelling activities can be conducted as table-top exercises, workshops, or formal detailed analysis. This flexibility in approach allows you to adjust to your time and budgetary constraints. What will however be a key element of all approaches is ensuring that the data flows, trust boundaries, and assumptions around security are all understood and can be articulated.

Based on our experience of conducting threat-modelling exercises on behalf of customers, we see the tremendous benefit of having people with an attacking mind, who have not previously been involved in the product, present. The goal here is not to antagonise, but rather to provide a fresh perspective with an experienced, attack-focused, mind. This approach will facilitate an almost chess-like game of threat modelling, with the attacker proposing a possible attack while the architects, designers, and development leads propose or describe existing mitigations in the design or planned implementation, or alternatively acknowledge it as a known risk. This process will ultimately lead to a comprehensive list of possible attacks, the design elements which mitigate them, or considerations and requirements for development to mitigate.

14.1.6 Cyber-Security Related Actions and Activities

So in summary, during the design, architecture, and technology stack selection phase the following cyber-security related activities should ideally occur:

- Hardware design and component selection decisions based on the required level of security and the considerations outlined in section 14.1.1.
- Software design decisions, component selection, and implementation planning based on the required level of security and considerations outlined in section 14.1.2.
- Review the proposed design and architecture taking into consideration the functional requirements that apply in section 14.1.3.

[18] http://www.amazon.co.uk/Threat-Modeling-Designing-Adam-Shostack/dp/1118809998

- Ensure there is understanding and reviews of security on a product, element, or device level as well as any integration with wider systems. Understanding the exposure that a device compromise would cause to these wider systems is an imperative.
- Conduct detailed threat-modelling activities at system, device, and functional requirement levels prior to implementation.

15 Phase 4: Implementation

The implementation phase naturally builds on the earlier work performed. The table below lists recommendations to ensure a secure implementation of software. In addition to these areas it is important that developers are trained so they have residual knowledge around secure software development.

Implementation Activity	Description	Reasoning
Adherence to secure programming guidelines outlined by the project	Developers should be trained in and adhere to secure programming guidelines. This is especially true for, but not limited to, unmanaged code such as C and C++.	Defining secure programming guidelines and ensuring all developers are aware of them will facilitate expedient peer code review while also reducing the introduction of common security mistakes.
Perform platform implementation and lockdown early on in development lifecycle	All base platforms should be configured early on for development to be as practically secure as possible and as representative of the final system as possible. This includes removing nonessential binaries, running processes with the intended privilege levels, configuration of sandboxing etc.	Developing on a system which is representative in terms of security from the outset will minimise the problems experienced during integration. If this is not done early on and these changes are made towards the end of development, the risk of fundamental implementation challenges will arise and the likelihood the original security requirements cannot be met without impacting delivery timelines is increased.
Use agreed developer tooling in the appropriate configurations.	All developers should use the agreed developer tooling in the appropriate security configurations.	Developing using tooling of the appropriate version in the correct configuration from the outset will minimise the problems experienced during later stages of development. If this is not done from the outset, development may be required to resolve non-security bugs or incompatibilities which become apparent when intended tooling and configurations are used.

Implementation Activity	Description	Reasoning
Perform static code analysis as close to the developer as possible	Ideally any static code analysis should be done as close to the developer or point of writing as possible. In order or priority this would be: • Development • Commit • CI or hourly/nightly build • Weekly build • Branch promotion • Production build	Performing static code analysis as close to development as possible reduces the cost of analysis and resolution, as it can be performed on an iterative basis by the developer writing that code currently. The risk of waiting for weekly, branch promotion or gold builds is that the developer may have long ago moved on to other areas of the code. This time lapse will require re-familiarisation with the code, unit testing, and functional testing, all increasing effort and time required. This is also important in agile environments where code refactoring may be high and thus doesn't lend itself to detailed manual analysis at logic gates in the development process.
Ensure the use of latest components	Ensure that all external libraries are the latest version and that they address all known security vulnerabilities.	If a third-party component is found to be vulnerable late in the development lifecycle and needs replacing it will likely require a full regression and functional testing cycle. Shipping a product with known-vulnerable third-party libraries is considered generally poor practice.
Positive and negative unit and functional test case creation	Ensure that both positive and negative test cases are developed for, including but not limited to authentication, authorisation, and traffic processing.	Developing negative test cases can uncover security issues very early on, while providing a robust regression test set going forward.
Security-focused manual code inspection	Based on the threat models and data flows, a security-focused manual code inspection should be performed by individuals highly familiar with software vulnerability discovery and exploitation	Independent analysis by individuals familiar with common language and complex logic vulnerability discovery can highlight potentially serious issues that may be overlooked by development.

15.1.1 Cyber-Security Related Actions and Activities

- Adherence to secure programming guidelines.
- Platform lockdown early on in the development lifecycle.
- Use of agreed developer tooling in defensive configurations.
- Static code analysis performed as close to development as possible.
- Ensuring latest versions which resolve known security issues of third party libraries and components are used.
- Production of positive and negative unit and functional test cases.

16 Phase 5: Verification and Testing

During the verification and testing phase a number of validation exercises will need to occur to verify that the requirements, design, architecture, and implementation all marry. Where they don't there should be clear justification as to why, or where no such justification does exist a bug report should be raised.

The recommended security-related verification and testing phase activities are as follows.

Verification and Testing Activity	Description	Reasoning
Production hardware schematic review and verification	Production hardware designs should be reviewed to ensure that any test functionality or interfaces have been removed prior to production run commencement.	Any production hardware with test interfaces or functionality present represents a potential security risk due to its ability to be used for research.
Base platform analysis	The base platform, including operating system, its core security properties and features, and its configuration, should be verified as being in line with the security requirements and no additional artefacts present.	Weak platform configuration can lead to compromise, privilege escalation, and persistence.
Network traffic analysis	All network traffic (wireless or wired) should be analysed to identify any interceptable, unencrypted, or modifiable data.	Any such cases can be cross-referenced with the existing risk analysis, requirements, design and architecture, data flow diagrams, and threat models to verify expected behaviour.
Interface analysis	All interfaces, both hardware and software, should be identified and verified as being expected.	Any identified interfaces that are not expected may represent a security risk.

Verification and Testing Activity	Description	Reasoning
Interface security analysis	All interfaces, both hardware and software, should be verified as having the required security	Any identified interfaces that are not protected in the expected way may represent a security risk.
Verification of functional security requirements (see section 13.2)	All high-level functional security requirements should be validated as being present and operational. They should also be subject to negative testing (subversion, fuzzing, and similar).	All security requirements should be validated as being present and operational while also being negatively tested to ensure effectiveness.
Verification of functional security design and architecture requirements (see section 14.1.3)	All design and architecture functional security requirements should be validated as being present and operational. They should also be subject to negative testing (subversion, fuzzing and similar).	All security requirements should be validated as being present and operational while also being negatively tested to ensure effectiveness.
Trust boundary review, functionality assessment and fault injection	All trust boundaries should be reviewed for functionality and subject fault injection, using negative test cases such as fuzzing.	All trust boundaries should be verified as being robust both in terms of supported and exposed functionality as well as being able to process malformed or unintended input without crashing or otherwise misbehaving.
Side channel attack defence verification	Where side-channel attack defences have been implemented either in hardware or software they should be verified.	Side channel attack defences are at risk of being undermined or otherwise impacted from hardware and software changes so should be verified as functioning as expected.
Targeted security focused code reviews	Security focused code reviews should be performed on the most sensitive or security impacting. These areas normally include the boot process, security enforcement, mitigations and similar.	Logic issues in such areas which are rarely detected through static analysis and sometimes difficult to discover during dynamic analysis could have wide ramifications on the security of the platform.

Verification and Testing Activity	Description	Reasoning
End-to-end functional security assessment, product penetration test, or code-assisted product penetration test	An assessment conducted by either an internal or external team of individuals with significant experience in solution and element level security assessments.	Independent validation by individuals with focused skills in attacking complex systems can highlight problems that were unconsidered or overlooked in any of the previous phases.
Backdoor identification	Attempt to discover deliberate and accidental backdoors created in hardware or software during later phases of development, testing, and bug fixing.	Backdoors, both accidental and deliberate, can have a very serious impact on product security.

16.1.1 Cyber-Security Related Actions and Activities

- Production hardware schematic review and verification.
- Base platform analysis.
- Network traffic analysis.
- Interface analysis.
- Interface security analysis.
- Verification of functional security requirements.
- Verification of functional security design and architecture requirements.
- Trust boundary review, functionality assessment and fault injection.
- Side channel attack defence verification.
- Targeted security focused code reviews.
- End to end functional security assessment or product penetration test.

17 Phase 6: Product Security Sustainment and Maintenance

Sustainment is one of the most overlooked phases and encompasses a whole set of policies, procedures, and technical activities.

A product sustainment plan typically needs to be able to:

- Receive and process reports of security issues from external parties.
- Proactively monitor for reports of security issues in third-party components used and work with development to integrate as appropriate.
- Regularly liaise with vendors of components used to identify if further releases have occurred that address security issues.
- Maintain a capability that can triage, resolve, test, ship, and distribute patches for security issues identified.
- Have a plan in place for worse case scenarios such as product recall or widespread repair.

This last point in particular is of key importance. We regularly see vendors who do not maintain such capabilities and as a result are unable to reasonably issue security patches even if there was a pressing need.

18 Conclusions and Summary

This paper has covered in both breadth and depth the cyber-security and privacy considerations that IoT implementers should consider during product conception and development. As previously discussed, and echoing the European Union, these areas should be considered early in product development. If this is not done there is the very real risk that the future systems will be as susceptible to attack as systems today, but with a larger number running more diverse platforms performing a wide array of functions. Also it is important to understand that while there is a lack of regulation the same cannot be said for the risk of litigation. As recent cyber-security compromises have shown, there is an increasing appetite for class action lawsuits and other forms of litigation in the event of a cyber-security breach. As a result there is a very real risk of financial penalty for designers, developers, manufacturers from security flaws, to say nothing of the cost of a product recall or repair.

We have also explained the practical activities that should be undertaken by those developing and testing IoT products. However, we also appreciate that to ensure such products are secure will require the entire supply chain to coordinate and collaborate, from silicon designers and manufacturers through to open-source projects that may underpin numerous parts of the IoT ecosystem. Only by making security a default consideration can the costs be driven down to manageable levels.

For modern systems to be securely designed, built, and sustained requires close collaboration across many phases and teams. Only by taking this holistic approach can vendors hoping to develop secure solutions do so in any meaningful manner.

19 Further Reading

The following are further sources of applicable information:

- Microsoft Security Development Life-cycle
  https://www.microsoft.com/security/sdl
- Threat Modelling: Designing for Security
  http://www.amazon.co.uk/Threat-Modeling-Designing-Adam-Shostack/dp/1118809998
- Building Security In Maturity Model
  http://bsimm.com/
- SafeCode Initiative
  http://www.safecode.org/
- CERT Secure Coding Guidelines
  https://www.cert.org/secure-coding/
- The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities
  http://www.amazon.co.uk/Art-Software-Security-Assessment-Vulnerabilities/dp/0321444426
- MITRE Common Weakness Enumeration
  https://cwe.mitre.org/
- The Open Web Application Security Project (OWASP)
  https://www.owasp.org/index.php/Main_Page
- European Union: IoT Privacy, Data Protection, Information Security Fact Sheet
  http://ec.europa.eu/information_society/newsroom/cf/dae/document.cfm?doc_id=1753
- CESG: Platforms Secure by Default
  https://www.cesg.gov.uk/publications/Documents/platforms_secure_by_default.pdf
- CESG: Secure by Default: Sensitive Data on Consumer Platforms
  http://www.cesg.gov.uk/publications/Documents/secure_by_default-sensitive_data.pdf
- CESG: End User Devices Security and Configuration Guidance
  https://www.gov.uk/government/collections/end-user-devices-security-guidance

- EU Internet of Things Initiative
    - http://www.iot-i.eu/public
- Fault attacks on secure chips
    - http://www.cl.cam.ac.uk/~sps32/ECRYPT2011_1.pdf

20 Thanks and Acknowledgements

The author would like to thank Matt Lewis, Richard Turnbull, Alex Brown, Aidan Marlin, Stephen Tomkinson, Richard Leach, Andrew Hickey, and Robert Horton of NCC Group for their peer review and feedback during the development of this paper.