Enforcing Policy-Based Security Models for Embedded SoCs within the Internet of Things

Matthew Hagan, Fahad Siddiqui, Sakir Sezer, BooJoong Kang, Kieran McLaughlin
Centre for Secure Information Systems (CSIT), Queens University Belfast
Belfast, United Kingdom
m.hagan, f.siddiqui, s.sezer, b.kang, k.mclaughlin@qub.ac.uk

***Abstract*—Within complex IoT ecosystems and network structures, hard to find vulnerabilities have potential to cause significant disruption and damage. In addition, device tampering and re-purposing can threaten business models of service providers. The vulnerability surface area of the ecosystem ranges across the entire system architecture, from the cloud to the IoT device. These can be introduced at any stage of the device life-cycle, including design, programming, manufacturing, integration, operation and maintenance of the device. While threat modelling during the design phase can alleviate some potential vulnerabilities, it is more difficult or even impossible to mitigate problems for devices already in the market. A policy-based device security model is proposed as an approach, that can be enforced using hardware and software security architectures.**

This paper reflects on existing literature on threat modelling and how derived security models can influence the design phase. This contribution proposes that by using the threat modelling to define specific use case security policies within the security model, OEMs will be able to tailor their solution to conform to the user's security requirements. Platform vendors, on the other hand, will have reduced design costs as they can offer generic solutions for differing levels of criticality.

An example scenario is provided using an industrial PLC as the attack target. While threat modelling can establish countermeasures for both the design process and policy defining, the policy can be introduced quickly, whereas the design method approach requires extensive modification to the system firmware.

***Index Terms*—Access Control, Attack trees SDLC, Secure by design, Security Modelling, Security Policy, SELinux, STRIDE, Threat Modelling, Trusted computing**

I. Introduction

The IoT market has grown significantly, with a massive range of IoT products available to consumers as well as enterprise and industry. Estimates from ARM predict that IoT will proliferate to a trillion devices by 2035 [1]. Simultaneously, it is well known that opportunities exist for adversaries to take advantage of IoT devices, exposing them to a myriad of security risks. The emergence of powerful embedded devices, deployed within the IoT and critical infrastructure to control critical tasks, poses new challenges for hardware providers, *Original Equipment Manufacturers* (OEMs) and users, who must enable security throughout the development and usage life-cycle, from product idea to decommission. Recent attacks and discovered vulnerabilities have shown that existing security mechanisms within IoT are either implemented poorly or have not been considered. As a consequence, devices have been found to be vulnerable in various ways, leading to attacks affecting provision of service, privacy, integrity, loss of sensitive information from an enterprise and even industrial applications.

The IoT ecosystem comprises of the following components:

- **Cloud:** Providing data storage and overarching computing services.
- **Gateway:** Interfacing the edge to the cloud component.
- **Edge:** The end-point device containing sensors and actuators

Each of these layers requires its own security considerations and faces different threats and attacks. Meeting the diverse security requirements of versatile IoT use-cases is also a major challenge, in particular since third party hardware and software components, used in devices, do not consider the end device's security requirements. The importance of integrating secure design practices within the IoT product development life-cycle is paramount for protecting against threats like counterfeiting, privacy attacks, system compromise and damage, all of which can cause severe reputation damage [2].

Devices may contain confidential data and control critical components of a process within the organisation, for example, using manufacturing data to actuate physical components on a factory floor, which could be targeted by an attacker. In the example of the Mirai bot-net attack [3], vulnerabilities within different IoT end-points with common software components were exploited to launch a distributed denial of service (DDoS) attack against cloud infrastructure. In light of such issues, IoT device manufacturers and OEM vendors must also now conform to an increasingly regulated environment [4], [5] with significant intervention and penalties where lax product security causes risk to consumers [6].

The widely adopted approach to discovering and mitigating security issues is to perform *Application threat modelling*. This is the process of assessing application and device functionality and identifying target components that an adversary may target [7] [8]. The security model, typically in the form of technical guidelines, is then used to implement technical countermeasures to anticipated threats. Threat modelling also complements the *software development life-cycle* by ensuring security is considered from initial development until decommission.

However, in practice, threat modelling can be compromised by corporate practices such as desire to minimise time-to-market and development costs. For example, during the

design phase, OEMs will likely integrate proprietary third party libraries, drivers, applications and hardware platforms. Usage of third party code is well known to be a potential security risk [9] and vendors may refuse to make source code and entire functionality available for security analysis to protect their intellectual property. Code reuse [10], software version fragmentation and enforced obsolescence [11] are other examples of where vendor practices may compromise consumers' desire for entire life-cycle security. Should a significant new threat emerge after deployment, that wasn't anticipated during the threat modelling phase, the OEM is particularly constrained in how they can alleviate the threat, since the threat modelling process performed produced a security model for the previously known threats. To address the new threats, an entirely new security model should be produced, which may be technically infeasible to implement. OEMs are therefore dependent on the platform vendors' assurances and abilities to secure the device at hardware and likewise for third party developers for software. Modelling and mitigation of threats is particularly important for OEMs, since the third party hardware and software components do not consider the end device's security requirements, which may vary widely between deployments.

Section II will explore existing mechanisms for determining and mitigating threats to IoT devices and systems during the product life-cycle. Section III will then observe related work, consisting of studies and applications of threat and security modelling within IoT and relevant environments. We will then present our proposed policy-based approach in Section III, followed by a realistic implementation scenario in Section Vto compare with the common guideline based approach. Section VI will wrap up the findings and propose future work in the area.

II. Background

A. Security within the Product Development Lifecycle

Microsoft's Security Development Life-cycle [12] details high level goals for achieving security. While these goals are designed for software development, they are relevant to IoT development. In particular, the *secure by design* and *secure by default* goals define the need to perform threat modelling. *Threat models* are commonly used to assist in determining the valuable assets of the system and possible attack scenarios that may occur. From this, a derived *security model* can provide guidelines for developers within the design process to secure the system's assets and circumvent the recognised threats.

To achieve secured design, the structure should be approached from a wholistic perspective, before separating into the components and looking at the attacker's perspective of accessing the system.

1) System based: By analysing the system from a holistic perspective, application and data flows can detail where vulnerabilities may lie or identify attack points. Data flow diagrams are useful high level tools for verifying design and production differences. Detailed data flow diagrams can also be used to identify trust boundaries and data flow issues [13].

2) Asset based: Identifying valuable, sensitive assets allows for prioritising protection measures. Attack tress, defined from the system perspective, can locate paths to the target asset.

3) Attack based: By studying the attacker's goals and methodologies, prioritisation of target areas and severity of attacks can be considered for mitigations.

Kohnfelder *et al.* [14] developed the attack based STRIDE and DREAD concepts of threat modelling at Microsoft. This work was an important milestone, differentiating between ad-hoc mitigations within the design and a formally produced threat analysis. Focusing on attack scenarios, STRIDE considers threat domains that attack consequences may fall under; ***S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service* and ***E**levation of privilege*. The DREAD concept attempts to quantify the threats in terms of factors like criticality and likelihood of occurrence. The acronym represents the terms ***D**amage potential, **R**eproducability, **E**xploitability, **A**ffected users* and ***D**iscoverability*.

B. Defining the Security Model

Fig.1 details the secure development and flow used for *Application threat modelling* to identify threats and define the *security model*.

As recognised by [15], threat modelling is largely a theoretical analysis task, dependent on human insights to identify and consider threat. In the traditional approach, a unique security model comprises in the form of system security requirements, which is derived from considered threats. It is then used to define the security architecture of the device, potentially for the entire life-cycle of the device. The security of the device or application is dependent on its enforced security model, which is the end-product of threat modelling. With risk of new threat and vulnerability discovery at any part of the device, this approach is not satisfactory. A change to the security model, based on a newly discovered threat, would require redesigning the system. In an optimistic scenario, this could be achieved with a software update. However, lower level changes, such as to hardware operation, are likely impossible to achieve once the device has been deployed.

C. Additional Techniques and Methodologies

Attack trees, defined by [16], [17] offer an alternative modelling approach, using an end goal perspective, from which the steps to achieve this are established. Each of these steps is represented by a leaf node, along with its subjective cost. Multiple or alternative prerequisites for tasks can be labelled using *AND* or *OR* nodes. While independent and more simplified than formal threat modelling methods such as STRIDE, attack trees may be considered a useful component of the threat modelling process, as they can establish and quantify, to an extent, likelihood of attack route, based on attacker cost.

In terms of implementation of policy within IT systems, the US government has applied focus to *Mandatory Access Control* (MAC) and *Discretionary Access Control* [18] [19],

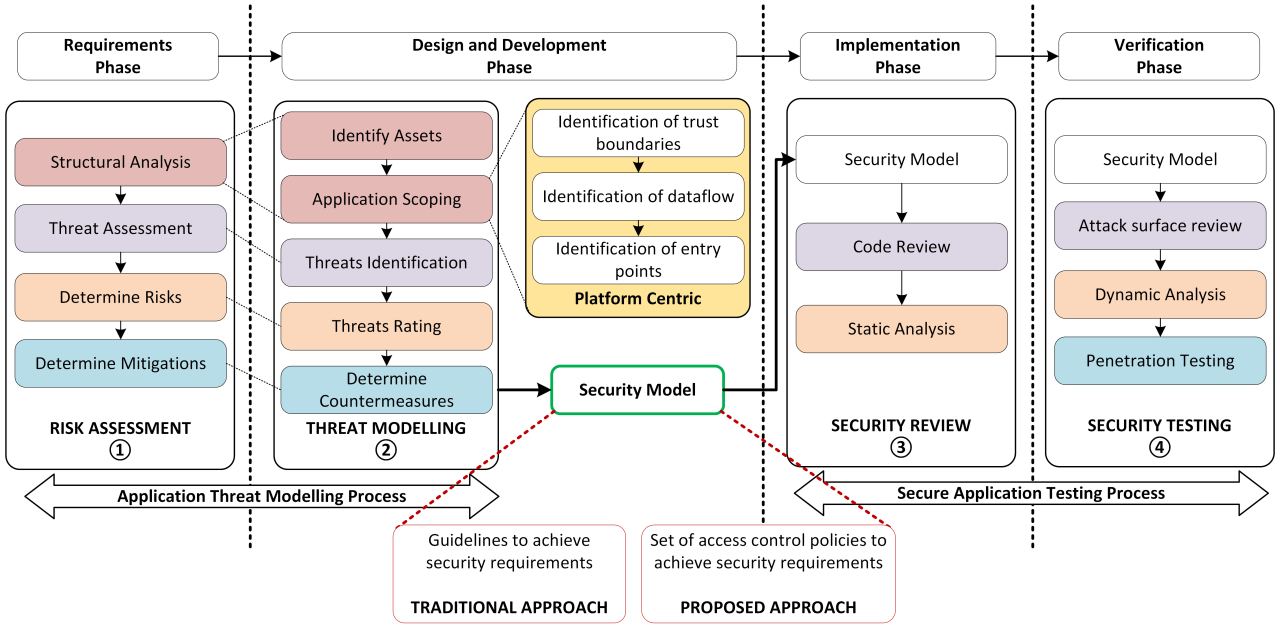

Fig. 1. The security model flow, from risk identification, modelled into threats and used to derive security model, for design level implementation.

culminating with the release of the open source *Security Enhanced Linux* (SELinux) platform, developed by the National Security Agency (NSA) [20]. SELinux is used to enforce access control policies within the operating system, enforcing the *Principle of least privilege* [21], confining user capability and separating processes to limit damage in case a breach occurs.

The concept of applying access control policy has further been explored within hardware. Siddiqui *et al.* [22] detailed a pro-active, policy based hardware enforcement mechanism known as a *Security Policy Engine*. This hardware based approach offers independent monitoring of transactions on the system communication bus, providing an additional layer of security that can protect the device when, for example, software based protections have been bypassed or compromised. Upon detection of a breach in the system, the *Security Response Engine* can take proactive countermeasures, such as deletion of secret data or disabling the device.

ARM [8] has developed a set of security models and specifications known as the *Platform Security Architecture* (PSA). This work aims to simplify and enhance the security evaluation process for IoT devices, making the development process more cost efficient and secure. The requirements to meet PSA's objectives are documented at a high level, applicable to developers within the entire ecosystem. The PSA is used as a guideline, documenting steps to create a threat model to define security objectives. From here security function requirements can be generated to counter threats. The PSA is platform agnostic, and also takes into consideration IoT applications where a full Trusted Execution Environment (TEE) is too large for the device.

III. Related-work

This section will survey existing implementations of security modelling.

Dorsemaine *et al.* [23] performed a threat analysis of attacks against a range of connected IoT thermostats, based on different characteristics of the devices and their interfaces. This work placed importance on considerations for interconnectedness of devices, where a device may be trusted by another device with a different security requirements.

Amthor [24] produced a uniform formal framework, capable of being adapted to different application domains. This work considered existing security analysis methods and their lack of adaptiveness to different application domains. Their work was demonstrated using an SELinux implementation.

Within cyber physical systems domain, Khan *et al.* [25] demonstrated the application of STRIDE to cyber physical systems. Although designed for software systems, STRIDE was found to be lightweight but an effective method for categorising recognised threats to the defined cyber physical system. Friedberg *et al.* [26] further described the combination of safety and security threat models within cyber physical systems, known as STPA-SafeSec. This work highlights dependencies between security vulnerabilities and system safety.

Saini *et al.* [27] used the concept of *attack trees* for threat modelling of attack scenarios against a web proxy service. This provided a qualatitive, practical approach to dealing with security risks, allowing analysts to make better informed choices when mitigating potential vulnerabilities.

Tan *et al.* [28] presented a decentralised system-level security approach, using resource isolation as the security foundation of the embedded architecture. Resource isolation

is defined in the form of access control policies classified into base, owner and shared permissions.

IV. Policy based Security Modelling Approach

The flow diagram in Fig. 1 shows a typical example of processes used to derive, design and implement security models, typically used in the product design phase. The *requirements phase* consists of documenting security requirements relevant to the use case and identifying assets within the system, such as secret data, along with scenarios where these assets may be threatened. High level mitigations can be defined that would limit or prevent capability of realising the threat.

The *design and development phase* applies the requirements information to the design process. This phase technically assesses the threats, identifying the assets involved and confirming their security relevance. Identification of trust boundaries, data flows and entry points further scopes the application of the security model to the end product. Use of threat rating methods like STRIDE and quantifying methods like DREAD are used to establish the impact of each identified threat and its consequence on the device's security, which enables prioritisation of the threat and creation of countermeasures. In addition, *attack trees* at this stage can demonstrate potential attack paths that may be used, which can greatly assist the identification of components requiring additional protections.

The traditional approach at this stage is to define technical guidelines, which can then be used within the *implementation phase*. These guidelines direct developers when navigating security sensitive areas of the device. However, issues are likely to occur where the implementer has less control over certain aspects, such as third-party IP hardware blocks and software. Since the OEM has less influence over the design process for these components, they are constrained in terms of how they mitigate potential vulnerabilities.

A. Proposed security modelling approach

Our contribution to this area is to propose a policy based security model that can be tailored to the user's security needs, to provide a flexible security model that is easily manageable and adaptable during the device life-cycle. The enforcement of the policies can be carried out by an operating system component, such as *SELinux*, or by an external hardware based entity such as the *Security Policy Engine*. In the event of attempted re-purposing or exploit against the device, an additional layer of hardware-based security mechanisms and the deployed mitigations can protect the device. The policy enforcement engine comprises of the following features:

- *Hardware*: Monitoring system-level communication between data and memory peripherals, enforcing the defined security policies for each slave.
- *Software*: Checking application permission boundaries and identifying anomalous behaviour.

By utilising policies to enforce these requirements, the OEM does not have to rely on third party vendors' security assurances. The enforcement of policies, derived from threat modelling of the system, assets and attack scenarios, can ensure that the device operates as intended by the OEM. In addition, development costs may be lowered for the hardware vendor if a generic platform can be distributed to multiple users, who can derive their own security policies based on the desired security requirements, using the familiar *Application Threat Modelling* approach presented in Fig. 1 and enforce them on the device. Finally, should the security requirements of the device change after production, for example, a new feature is activated on the device or a new vulnerability is discovered, the OEM can distribute a policy definition update. For this approach to work, the vendor will be required to integrate a policy enforcement mechanism, such as SELinux or as proposed by other research.

V. Application use-case: Industrial Control System

This example scenario considers a typical, IEC 61131-3 compliant *Programmable Logic Controller* (PLC), that has been connected to a organisation's internal manufacturing network, from which it is used to control an industrial process.

A PLC has been chosen to apply this vulnerability scenario due to its versatility in function and generic nature. A PLC can be used to control a wide range of different purposes within industrial control systems, including having complete control over a process and being isolated from other parts of the network, or, more commonly, being used within a Distributed Control System with connectivity to the management network as shown in Fig. 2. PLCs typically maintain connectivity through Ethernet or serial devices, allowing interaction with a vast range of industrial hardware as well as standard networking connectivity. Each of these connectivity scenarios faces different security challenges [29]. A threat model will be created, using the supplied criteria, with consideration to both the infrastructure surrounding the PLC, as well as within the PLC itself.

A. Device attack scenario

The attacker can view the PLC from a variety of perspectives. The PLC itself may contain valuable information, such as proprietary information on processes. It's connection to valuable hardware, such physical actuators, yields is a high value target. Alternatively, the PLC's generic nature may also yield the device as a platform for an attacker to use to attack other parts of the network. Being in a critical location, the device will likely have access to other critical control processes apart from those directly connected, as detailed in Fig. 2. There may be further circumstances where the PLC is not directly compromised, but carries data and commands issued from a compromised device. Finally, the PLC, particularly in conditions where internet access is available, may be compromised by a generic malware threat, causing the PLC to become part of a botnet to launch attacks such as DDoS, as was the case with the Mirai bot-net.

B. Device attack surface

The following are classified in to three categories as listed in Table. I:

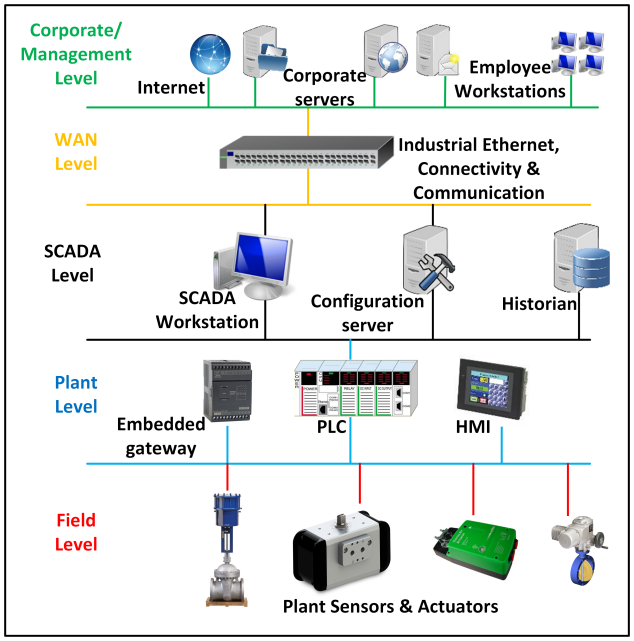

Fig. 2. A typical industrial network showing different levels of operation, with connectivity to industrial endpoints, and back to the corporate network.

1) Device centric attacks: These consider where the PLC itself is directly targeted in order to compromise PLC components or its directly connected controller or actuators.
2) Proxied Device attack: This is where a PLC vulnerability is exploited, but where the target device or data is not directly connected. Advanced Persistent Threats (APTs) and automated bots are examples of attacks.
3) Device passive attacks: These consider where the PLC is neither the target, nor the attacked device, but is directly involved in the attack process by handling information between the attacker and target.

C. Threat Analysis and Security Modelling

Three different attack categories are considered. For each category an analysis will be performed based on the flow diagram, with attack scenarios considered using STRIDE and quantified using DREAD. Table I shows the results of this process. This paper will go through the process of identifying appropriate policy based mitigations for one of these attack scenarios.

D. PLC centric attack

This example considers a multi-staged attack, where an adversary, with physical access to the device, puts the device into a failsafe mode to flash a signed, but vulnerable older firmware image through a diagnostic port. The device is already in deployment.

1) Risk Assessment: The inclusion of diagnostic ports on generic industrial devices is expected for customers wishing to diagnose faults. Legitimate reasons exist to permit firmware update through this method, such as where the device is

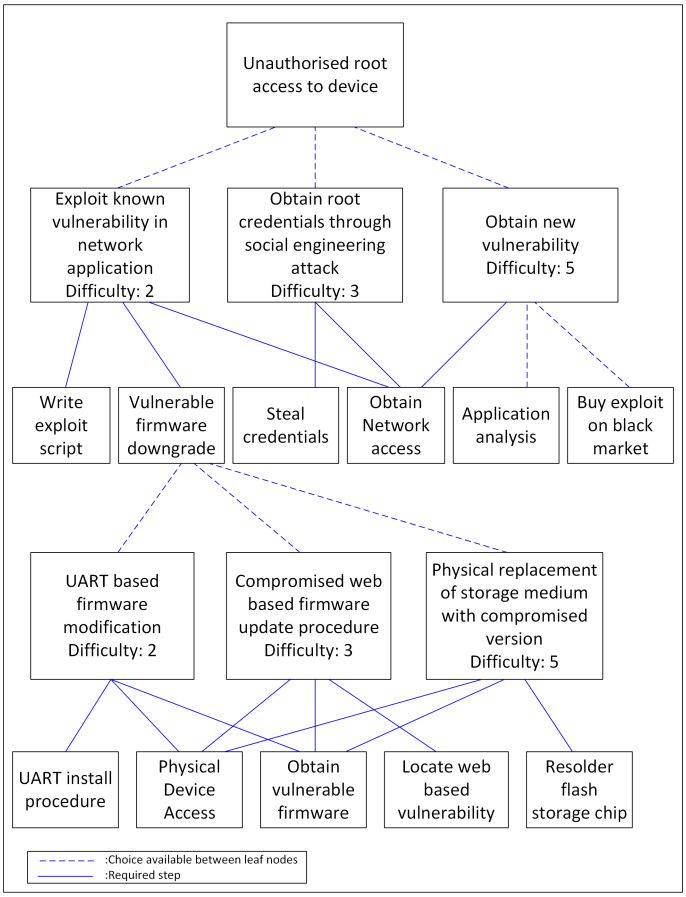

Fig. 3. Attack tree detailing possible steps for achieving unauthorised root access by combining a web exploit with a vulnerable firmware version downgrade.

standalone without network connectivity. The use of cryptographic signatures means that only authorised firmware can be used, but use of different firmwares used between customers and downgrade for compatibility reasons means firmware revocation has not been implemented. The mitigations in this case are to require physical access to the device and a signed firmware image only.

2) Threat Modelling: The overall target asset in this case is the firmware and operating system, as listed in I, which can be compromised when exploiting a vulnerability in an older firmware version. Within this attack there are two stages, detailed as follows;

1) *Firmware downgrade via UART*: Failsafe boot mechanism triggered within the bootloader software environment allows.
2) *Vulnerable application exploit via network*: Unauthorised device access gained after network based compromise of old firmware version.

The attack tree detailed in Fig. 3 details potential steps the attacker may take to achieve the respective goals of downgrading the PLC firmware and obtaining root access to the device. Within a full threat model, the attack tree may be extended to fully evaluate all methods to achieve the root node,

TABLE I
STRIDE AND DREAD ANALYSIS OF MODELED SOFTWARE AND HARDWARE ATTACKS AGAINST INDUSTRIAL CONTROL SYSTEM.

Class.	Target System	Target Asset	Detect.	STRIDE*	DREAD (Avg.)
Device Centric	**Firmware Storage**	Use of fail-safe method to downgrade to vulnerable firmware version	H	STRDE	9,5,1,8,5 (5.6)
	Network Peripheral	Embedded application exploit to gain unauthorised access	S/H	TID	7,8,7,8,9 (7.8)
		DOS Attack to exhaust network connectivity	S/H	TID	5,7,9,7,9 (7.4)
		Malware interaction with TCP/IP offload to corrupt packets	S/H	TD	8,3,2,7,2 (4.4)
	Memory	Exfiltration of memory contents over network	S/H	TRID	6,2,2,8,3 (4.2)
	Physical Casing	Removal of casing, side channel CPU attacks and probing of components	H	TRID	9,4,1,8,1 (4.6)
	Programmable Storage	Exfiltration of process application over network	S/H	TRID	8,4,2,8,2 (4.8)
	Serial Peripherals	DOS attack against industrial process	S/H	TID	6,5,5,6,7 (5.8)
		Injection of faulty data to corrupt industrial process	H	TID	9,4,3,8,4 (5.6)
Device Proxied	**Corporate Network**	APT malware installed after remote access exploit to find and exfiltrate corporate data from network.	S/H	IE	9,1,2,9,6 (5.4)
	Remote Host	Botnet style DOS attack against remote host	S/H	D	5,6,7,4,5 (5.4)
Device Passive	**Industrial Process**	Active probing of sensors/actuators to reverse engineer process	S/H	STI	8,2,5,8,6 (5.8)

STRIDE: S=*Spoofing* ; T=*Tampering* ; R=*Repudiation* ; I=*Information Disclosure* ; D=*Denial of service* ; E=*Elevation of Privilege*.
DREAD: D=*Damage* ; R=*Reproducibility* ; E=*Exploitability* ; A=*Affected Users* ; D=*Discoverability*.
Detect: S=*Software level detection* ; H=*Hardware level detection*.
* STRIDE features present.

that is the adversary's goal. The aim of the attack tree defining process is to establish plausible routes an adversary may take, calculating the cost and complexity at each step. Used in conjunction with STRIDE, attack trees may strengthen the modelling process, leading to a more robust security model.

For the installation procedure the application boundaries are between the physical entry point (UART port), the bootloader and the firmware upgrade procedure. For the network exploit, the application boundaries are the network interface, the operating system (network driver) and the vulnerable application. To evaluate the attack's potential to damage the overall system, STRIDE based threat identification of the problem locates the possible threats.

- *Spoofing*: Although possible, this is outside attack scope.
- *Tampering*: Tampering with device firmware and process data.
- *Repudiation*: Log deletion not within scope of the attack.
- *Information Disclosure*: Sensitive process information will be disclosed to attacker.
- *Denial of Service*: Attack aim is to compromise running service but not disrupt.
- *Elevation of Privilege*: Root access is achieved after vulnerability exploit.

DREAD scoring of the identified threats considers the severity of a successful exploit that can damage the system.

- *Damage*: **8** - Damage to costly industrial process likely.
- *Reproducability*: **6** - Requires physical access to device and correct tools to install.
- *Exploitability*: **6** - Multi-step attack process using physical access and network access.
- *Affected users*: **8** - Process may be halted or disrupted. Downtime may severely affect users.
- *Discoverability*: **5** - Knowledge of the vulnerability requires research of the device, firmware, and applications.

E. Determining Countermeasures

Having performed the risk assessment and threat modelling phase as detailed in Fig. 1, the final stage is to define the security model that can be used to defend against the attack. This section will compare the normal approach to our proposed policy-based approach.

1) Guideline based security model: The traditional approach to this problem involves generating guidelines to realise and enforce security model during development, which provides security only to attacks covered during the development phase. Considering that the device is already deployed, the OEM is limited in how it can create robust countermeasures to prevent the attack. The manufacture will be limited to issuing a software update, usage guidelines, or using the information for refining a future product. Based on this information, the following are example guidelines for consideration, that could be used in this case;

- *Hardware Development*: Consider case for UART disabling in future firmware.
- *Software Development*: Implement and enable firmware revocation in future firmware.
- *Software Development*: Authenticate bootloader with logging capability.
- *Usage Guideline*: Allow only trusted personnel access to device and keep records.
- *Usage Guideline*: Ensure firewalling and Intrusion Detection System rule to detect network attack against PLC.

In the case of the technical guideline, removal of the UART and introduction of firmware revocation versions may cause additional problems for other users who require diagnostics or have stringent software version requirements. A configurable bootloader with an authentication process would require a more difficult update process and would require specifying guidelines for password management by the customer and organisation. Any failsafe password mechanisms may further be compromised in the future.

2) Proposed policy based security model: The proposed policy-based security model approach derives policies which can be managed and modified to deploy different countermeasures to curtail existing or future threats. Since the attack scenario is multiphased (firmware downgrade followed by vulnerability exploit), countermeasures addressing either of these points will thwart the attack. This is particularly important as

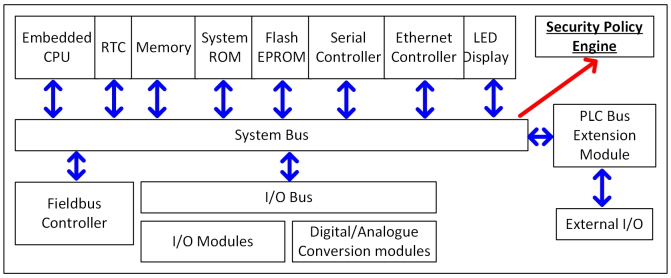

Fig. 4. PLC Architecture detailing location of policy enforcement engine

the two phases are distinct and unrelated, from a functionality context. The policies that could be implemented are as follows:

1) *Monitoring UART firmware update*: By monitoring the UART peripheral and firmware's flash memory storage for write attempts, the unauthorised firmware update process can be blocked at hardware level. Other debugging activity would remains permitted.
2) *Enforcing access control policies using SELinux*: An application control policy in SELinux could detect exploitation and attempted access to unauthorised areas by the application. At hardware level, the application's access to memory regions outside of those authorised could be detected. The network based exploit and subsequent control of the system would trigger behavioural based policies in this case.

Both attack phases may be handled separately prevented through use of policies, determined from the scenario and threat modelling. Our proposed approach provides an additional layer of security that can mitigate the existing vulnerabilities, without requiring redesigning the system architecture. The OEM, in this case, could devise the policy update, which could be applied without affecting the operating system or core functionality of the device. Fig. 4 demonstrates the PLC architecture with the potential location of the policy enforcement engine.

VI. Conclusion and Future Work

This paper's contribution has been to present the current problems in existing approaches to device security modelling, in that they are restricted to influencing design and development stage and not after deployment, when changes to the device are more difficult to implement. This paper has explored the possibility of directly deriving and applying policies from threat analysis. This is particularly useful where the hardware vendor wishes to lower costs and provide a generic platform for multiple use cases, and where a flaw has been found in a device is already released in the market that could not be rectified without redesign.

This contribution has walked through a realistic use case that shows how the application of policies can mitigate security flaws, with the alternative solution being a complex upgrade procedure that customers may be unwilling to perform. In addition, the threat modelling process has been stepped through, detailing a workable process that can generate a usable technical policy to protect an asset from potential threats.

This work aims to complement both existing work into threat models along with security modelling applications in software and hardware. The next phase of this research is to further refine the approach of deriving a technical policy security model from a high level security problem. This may involve a combination of existing threat analysis methodologies. Additionally,a low level implementation of the proposed approach into a generic hardware platform will be performed to evaluate its effectiveness for systems with differing criticality.

References

[1] "The route to a trillion devices: The outlook for IoT investment to 2035," ARM, Tech. Rep., 2017. [Online]. Available: https://community.arm.com/cfs-file/__key/telligent-evolution-components-attachments/01-1996-00-00-00-01-30-09/Arm-_2D00_-The-route-to-a-trillion-devices-_2D00_-June-2017.pdf

[2] "Future-proofing the Connected World," IoT Working Group, Cloud Security Alliance, Tech. Rep., 2016. [Online]. Available: https://downloads.cloudsecurityalliance.org/assets/research/internet-of-things/future-proofing-the-connected-world.pdf

[3] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 1093–1110. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

[4] U.S. Department of Homeland Security, "Strategic Principles for Securing the Internet of Things," US Government, Tech. Rep., 2016. [Online]. Available: https://www.gov.uk/government/publications/national-cyber-security-strategy-2016-to-2021

[5] Department for Digital, Culture, Media & Sport, "Secure by Design: Improving the cyber security of consumer Internet of Things Report," HM Government, UK, Tech. Rep., 2018. [Online]. Available: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/686089/Secure_by_Design_Report_.pdf

[6] "FTC Approves Final Order Settling Charges Against TRENDnet, Inc." US Federal Trade Commission, Tech. Rep., 2014. [Online]. Available: https://www.ftc.gov/news-events/press-releases/2014/02/ftc-approves-final-order-settling-charges-against-trendnet-inc

[7] D. Lowry, E. Keary, and D. Rook, "Application Threat Modeling," OWASP, Tech. Rep., 2015. [Online]. Available: https://www.owasp.org/index.php/Application_Threat_Modeling

[8] Arm Ltd., "Arm platform security architecture overview," 10 2017, revision 1.1. [Online]. Available: https://pages.arm.com/PSA-Building-a-secure-IoT.html

[9] E. Derr, "The impact of third-party code on android app security," in *Enigma 2018 (Enigma 2018)*. Santa Clara, CA: USENIX Association, 2018. [Online]. Available: https://www.usenix.org/node/208134

[10] "TPLINK TLWR740N ROUTER REMOTE CODE EXECUTION," Fidus Information Security, Tech. Rep., 2017. [Online]. Available: https://www.fidusinfosec.com/a-curious-case-of-code-reuse-tplink-cve-2017-13772-v2/

[11] X. Zhang, Y. Zhang, J. Li, Y. Hu, H. Li, and D. Gu, "Embroidery: Patching vulnerable binary code of fragmentized android devices," in *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sept 2017, pp. 47–57.

[12] S. Lipner and M. Howard, "The trustworthy computing security development lifecycle," Microsoft Inc., Tech. Rep., 2005. [Online]. Available: https://msdn.microsoft.com/en-us/library/ms995349.aspx

[13] M. Abi-Antoun, D. Wang, and P. Torr, "Checking threat modeling data flow diagrams for implementation conformance and security," in *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '07. New York, NY, USA: ACM, 2007, pp. 393–396.

[14] M. Howard and D. Leblanc, *Programming Multicore and Many-core Computing Systems*. Microsoft Press, 2003.

[15] S. Ray, E. Peeters, M. M. Tehranipoor, and S. Bhunia, "System-on-chip platform security assurance: Architecture and validation," *Proceedings of the IEEE*, vol. 106, no. 1, pp. 21–37, Jan 2018.

[16] B. Schneier, "Attack trees," in *Dr. Dobb's Journal*. CMP Media, Inc., 1999. [Online]. Available: https://www.schneier.com/academic/archives/1999/12/attack_trees.html

[17] B. Schneier, *Secrets & Lies: Digital Security in a Networked World*. New York, NY, USA: John Wiley & Sons, Inc., 2000.

[18] National Computer Security Center, *A Guide to Understanding Security Modeling in Trusted Systems*. DIANE Publishing Company, 1993.

[19] S. Chokhani, "Trusted products evaluation," *Commun. ACM*, vol. 35, no. 7, pp. 64–76, Jul. 1992.

[20] "Security enhanced linux," National Security Agency — Central Security Service, Tech. Rep., 2008. [Online]. Available: https://www.nsa.gov/what-we-do/research/selinux/

[21] J. H. Saltzer, "Protection and the control of information sharing in multics," *Communications of the ACM*, vol. 17, no. 7, pp. 388–402, July 1974.

[22] F. Siddiqui, M. Hagan, and S. Sezer, "Embedded policing and policy enforcement approach for future secure iot technologies," in *Living in the Internet of Things: Cybersecurity of the IoT - 2018*, March 2018, pp. 1–10.

[23] B. Dorsemaine, J. P. Gaulier, J. P. Wary, N. Kheir, and P. Urien, "A new approach to investigate iot threats based on a four layer model," in *2016 13th International Conference on New Technologies for Distributed Systems (NOTERE)*, July 2016, pp. 1–6.

[24] P. Amthor, "A uniform modeling pattern for operating systems access control policies with an application to selinux," in *2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)*, vol. 04, July 2015, pp. 88–99.

[25] R. Khan, K. McLaughlin, D. Laverty, and S. Sezer, "Stride-based threat modeling for cyber-physical systems," in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, Sept 2017, pp. 1–6.

[26] I. Friedberg, K. McLaughlin, P. Smith, D. Laverty, and S. Sezer, "Stpa-safesec: Safety and security analysis for cyber-physical systems," *Journal of Information Security and Applications*, vol. 34, pp. 183 – 196, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214212616300850

[27] V. Saini, Q. Duan, and V. Paruchuri, "Threat modeling using attack trees," *J. Comput. Sci. Coll.*, vol. 23, no. 4, pp. 124–131, Apr. 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1352079.1352100

[28] T. Benjamin, B.-A. Morteza, and S. Zoran, "Towards decentralized system-level security for MPSoC-based embedded applications," *Journal of Systems Architecture*, vol. 80, pp. 41 – 55, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1383762117300322

[29] K. A. Stouffer, J. A. Falco, and K. A. Scarfone, "Sp 800-82. guide to industrial control systems (ics) security: Supervisory control and data acquisition (scada) systems, distributed control systems (dcs), and other control system configurations such as programmable logic controllers (plc)," National Institute of Standards & Technology, Gaithersburg, MD, United States, Tech. Rep., 2011.