Security-Minded Verification of Space Systems

Carsten Maple
WMG, University of Warwick,
Coventry, CV4 7AL
cm@warwick.ac.uk

Matthew Bradbury
WMG, University of Warwick,
Coventry, CV4 7AL
M.Bradbury@warwick.ac.uk

Hu Yuan
WMG, University of Warwick,
Coventry, CV4 7AL
H.Yuan.4@warwick.ac.uk

Marie Farrell
Department of Computer Science,
University of Liverpool,
Liverpool, L69 3BX
Marie.Farrell@liverpool.ac.uk

Clare Dixon
Department of Computer Science,
University of Liverpool,
Liverpool, L69 3BX
cldixon@liverpool.ac.uk

Michael Fisher
Department of Computer Science,
University of Liverpool,
Liverpool, L69 3BX
mfisher@liverpool.ac.uk

Ugur Ilker Atmaca
WMG, University of Warwick,
Coventry, CV4 7AL
Ugur-Ilker.Atmaca@warwick.ac.uk

***Abstract*—Modern space systems are increasing in complexity. The advent of the Internet of Space Things, coupled with the commercialisation of space has resulted in an ecosystem that is difficult to control and brings about new security challenges. In such critical systems, it is common to conduct verification strategies to ensure that the underpinning software is correct. Formal verification is achieved by modelling the system and verifying that the model obeys particular *functional* and *safety* properties. Many connected systems are now the target of a variety of threat actors attempting to realise different goals. Threat modelling is the approach employed to analyse and manage the threats from adversaries. Common practice is that these two approaches are conducted independently of one another. In this paper, we argue that the two should be mutually informed, and describe a methodology for security-minded formal verification that combines these analysis techniques. This approach will streamline the development process and give a more formal grounding to the security properties identified during threat analysis.**

TABLE OF CONTENTS

1. INTRODUCTION

Space systems are an essential part of national and international infrastructure for the purposes of data communication [1], environment monitoring [2], scientific experiments, global positioning, navigation [3], timing, and many others. Currently, the space sector is experiencing substantial change due to the advancement of technologies such as artificial intelligence (AI) [4] and the Internet of Things (IoT) [5]. Ensuring that these complex systems operate as they are intended is becoming increasingly challenging and important.

978-1-7281-2734-7/20/$31.00

Verification of complex systems remains a significant challenge. It is recognised that formal verification of complex systems in open contexts [6] is beyond current techniques and resources, fundamentally due to state explosion. On the other hand, informal methods such as simulation, have also long been recognised to have limitations; in particular, their "effectiveness in finding corner-case bugs significantly decreases" [7] over time. As such, hybrid techniques have been proposed and used in research and practice for several years [8]. These methods involve a mixed approach of formal and informal methods for verification, to provide improved levels of confidence in the system. For example, in [9] different verification and validation techniques are applied to a robot co-operative manufacturing task, in particular, formal verification (model checking), simulation-based testing, and user validation with experiments with a real robot. The results from verification with one technique are used to improve the inputs to the verification process or to guide the analysis for the other techniques thus improving confidence in the system. Given that such informal verification methods cannot cover all states of the system it is important to consider which states should be prioritised to provide adequate confidence in the correct functioning of the system.

Space systems have previously been developed by a small, closely controlled set of organisations, but, reduced costs and access to technology [10] has led to a much wider ecosystem with a more diverse supply chain [11]. This has, in turn, led to a reduction in the ability for control and oversight, as well as an increase in complexity. More commercial-off-the-shelf components are now "commonly used to build small satellites at the lower end of the cost range" [12]. Coupled with this is an increase in the ability and attractiveness of attacking space systems [13]. While verifying system operations in open (real-world) environments is a challenge, introducing an adversary into the consideration raises the challenge by at least an order of magnitude.

To address the challenge of assuring complex space systems in the presence of an adversary we present a novel methodology that we term *security-minded verification*. In order to present the concept, we discuss formal and informal techniques for verification before undertaking a threat modelling exercise to identify the threat actors, motivations and methods used to compromise space systems. We then present a methodology that combines the two approaches which results in the verification strategy being evaluated against threat modelling, and conversely using threat modelling to inform the strategy for

verification. The method provides the opportunity to redefine the requirements of the system based upon the results of the threat modelling.

We illustrate the methodology by discussing the scenario of cooperative autonomous docking performed by two robots in orbit (such as in order to perform orbital construction or on-orbit servicing). Typically these robots may be considered as cooperative, non-cooperative or semi-cooperative [14], and requirements may be developed to consider the behaviours of the robots in each of these cases. Rather than modelling and verifying the system without considering an attacker, we consider the changes to the specification in the case that an attacker can control one of the robots so that it is anti-cooperative. This leads to a change in the requirements and specification before we then undertake the verification strategy.

To summarise, the main contribution of this paper is to present an enhanced methodology for security-minded verification, based upon previous work presented in [15]. The concept presented in the previous work is enhanced by presenting a methodology that is more detailed and integrated. To illustrate the methodology, we present its application to an example currently being examined by the Future AI and Robotics for Space (FAIR-SPACE) Hub[2]. In this context, we introduce the novel definition of an anti-cooperative robot, that is a robot that has been compromised and is under the control of an adversary, to complement the definitions of cooperative, non-cooperative and semi-cooperative robots.

The remainder of this paper is structured as follows, in Section 2 we present related work on formal verification and security analyses of space systems. Section 3 presents the two existing individual methodologies for formal verification and threat modelling and Section 4 presents our proposed combined methodology. In Section 5 we present the use case of autonomous docking and apply this methodology to it. A discussion and future work are provided in Section 6. Finally, Section 7 concludes.

2. RELATED WORK

This section summarises pertinent related work for this paper and comprises subsections concerning formal verification of space systems, threat modelling of space systems and the combination of formal verification and cyber security.

Formal Verification of Space Systems

A recent survey of formal specification and verification techniques for autonomous robotic systems has revealed that, although there are many tools and techniques available, improvements are still required for their successful deployment in large, complex and autonomous systems [16, 17]. There has been much interest in autonomy from the space industry to support missions in these remote and dangerous environments because remote operation may be problematic or impossible due to distance, time lags or communication issues and manual operation may be hazardous or impossible. Additionally, autonomy can save time by removing the need for human intervention as well as help mediate failures. Due to the importance of many space systems and the services that they provide, it is vital that they are correctly and robustly verified. There are many approaches to formally verify such systems and we describe some of these in what follows.

[2]https://www.fairspacehub.org

Brat et al. have proposed an approach for the verification of autonomous space systems that is based on an assume-guarantee framework [18]. Their work is relevant for systems that are composed of 3 layers (planning, execution and functional layers) and they make use of model-checking, static analysis, synthesis and testing to demonstrate their approach. In other work, the authors compared tools for static analysis, model-checking and runtime verification against traditional testing of rover flight software [19]. Each of these formal techniques outperformed testing when locating concurrency errors.

Alves et al. have used UML-based formal assertions and runtime monitoring to verify and validate a Brazilian satellite launcher flight software [20]. The data for the runtime monitoring JUnit tests was collected from log files. Also relating to runtime monitoring, the R2U2 tool (Responsive, Realisable, Unobtrusive Unit) has been used in the development of small satellites [21, 22].

Event-B specifications have been combined with probabilistic properties to derive reconfigurable architectures for an on-board satellite system [23]. In this work, the models of reconfigurations are checked using PRISM for both the derivation (via refinement) of the system from its specification and the probabilistic assessment of their reliability and performance.

Cyber Security for Space Systems

The space systems and services currently being developed and deployed are highly dependent on *cyber* technologies, including software, hardware and other digital components [24]. This provides threat actors with a large surface to attack space systems. In addition to this increased surface, attackers are enjoying cheaper, more widely available and more effective tools with which to attack these systems. For example, Software Defined Radios (SDRs), are rapidly decreasing in cost and this thus the technology reduces barrier to entry for interaction with satellites [25].

Housen-Couriel [26] reviewed the satellite communication cyber attack conducted by the so-called Turla group, in the context of new threats to the space systems and legal responses in terms of international law. According to Kaspersky, the Turla group was more hazardous and harder to distinguish since they not only used complex and sophisticated resources but also implemented an exquisite satellite-based command-and-control system in their attack [27].

Livingstone and Lewis [13] have analysed the dependencies of the wide range of critical infrastructures on space systems, and how cyber security vulnerabilities pose serious and broad risks for these critical infrastructures. The authors have also identified future trends and changes in the space ecosystem, and examined several potential types of cyber attacks including an analysis of the impact if exploited. As in [26], they have also recommended cooperation in international law for the cyber security of space systems. The link between satellites and critical infrastructures is further emphasised in [28], with particular reference to the scope of NATO's missions.

In [29], the author analyses the current state of the industry and existing security practices. They recommend that several policy changes should be made to improve the state of space cyber security. Recommendations include that space organisations should be more open concerning working with cyber security researchers and facilitating information sharing. The latter is a particularly important issue in the evolving space sector, in which new entrants are unlikely to have the

cyber and information security expertise and knowledge of more established organisations.

The Centre for Strategic & International Studies (CSIS) assessed the threats to space systems in [30]. In this report, the threats of space are classified as three factors: (i) physical (e.g., kinetic and non-kinetic), (ii) electronic (e.g., jamming and spoofing radio communications) and (iii) cyber. The authors distinguish electronic attacks from cyberattacks in which the target is the data and components where this data is flowing through. The report explains how data traffic patterns can be used to monitor information itself as well as to insert false or corrupted data into the system. It is argued that such cyber attacks on space systems may cause data loss, system disruption or even permanent loss of a satellite. The report also contains details of geopolitical rivals to the USA with capabilities of launching a space attack.

The Consultative Committee for Space Data Systems (CCSDS) also published a report on current threats of space cyber security [31]. In the report, the threat actors and the types of threats they hold are discussed. The threats in different scenarios are listed with assessments of the impact if these threats are realised. The authors state that the threats identified, impacts and mitigation will change when considered under different scenarios compared to the hypothetical scenario presented.

Combining Formal Verification and Cyber Security

Clearly, security and formal verification are important aspects to providing robust space systems in the presence of an adversary. However, it is not clear that there is a comprehensive understanding of these requirements across all organisations involved in the space sector. Having recognised the need and challenge of security and formal verification of space systems, we believe that an appropriate methodology is required to ensure that these issues can be considered efficiently and robustly.

It is recognised that cyber security is often "bolted on as an afterthought" and that "this is not effective in practice" [32] and indeed "dangerous" [33]. To overcome challenges in the efficacy of formal verification and security, we propose a methodology that combines the two processes: threat modelling and formal verification. We call this approach *security-minded verification*. Our previous work, presented in [15], on using threat analysis techniques to guide the formal verification effort of the Cooperative Awareness Messaging (CAM) protocol (a common protocol used in autonomous vehicle communications) provides a basis for our security-minded verification methodology presented herein. In particular, here we provide a more detailed methodology rather than a concept, and a case study from the space domain to illustrate our approach.

There has been much work on applying formal verification techniques to analyse and verify the security aspects of particular systems as well as cryptographic protocols [34, 35, 36, 37, 38, 39]. However, such approaches are techniques to verify that the security requirements of a system are met and do not inform the verification process for the whole system. The closest approach to combining security and verification is perhaps that in which machine learning techniques are used to extract finite state machines from bank cards which implement variants of the EMV (Europay-MasterCard-Visa) protocol suite [40, 41]. These finite state machines can then be used for security analysis of the implemented protocol. However, this approach falls short of what we propose here, and we have not found any related work that describes a methodology for linking these two techniques, which is independent of the tools or approaches used for threat analysis and formal verification.

3. CURRENT METHODOLOGIES

In this section, we describe the current methodologies for both formal verification and threat modelling. Figures 1 and 2 illustrate these methodologies, respectively, in terms of a modified workflow diagram, with ovals representing the start and end points, rectangles representing tasks, diamonds representing decisions, and a rhombus (custom element) indicating the output from a task. Tasks and outputs have been coloured differently to aid in understanding how the two methodologies are combined. Verification tasks have been coloured green and outputs have been coloured lilac, threat modelling tasks have been coloured orange and outputs have been coloured blue, finally, tasks common to both have been coloured yellow and output common to both have been coloured pink. This is outlined in Table 1.

	Tasks	Decisions	Outputs
Verification			
Threat Modelling			
Common to both		N/A	

Table 1. Key to Figures

Formal Verification

In this subsection, we describe the process for formal verification that is illustrated in Figure 1. Formal verification is conducted as part of the software development process, ideally beginning before the system is implemented. We begin by defining the system under development. This may involve the development of software requirements documents or analysing existing documentation to provide the developer with a detailed understanding of the system to be developed.

Once the system has been defined, a formal model of the system is then constructed and the properties to verify are formalised. One approach would be via model-checking [42, 43], where a model of the system is created, usually, as a state transition system which is encoded as input to a model checking tool. Properties derived from the requirements document are encoded using a logic, such as Linear Temporal Logic (LTL) [44].

A model-checking tool can be run to demonstrate that the property holds on all runs through the transition system from an initial state. An example of this would be to build a model of the system using Promela, the input language to the SPIN model checker [45] and then to formalise the properties to be verified using LTL. Alternatively, we may wish to define the model and properties in the same language. This could also be performed with some form of temporal logic, for example, LTL or the branching time temporal logic Computational Tree Logic (CTL) [46]. Here both the model of the system and the property are defined in the same logic and proof is carried out within that logic using temporal proving tools for that logic, for example, LTL theorem proving [47] via TRP++ [48] or

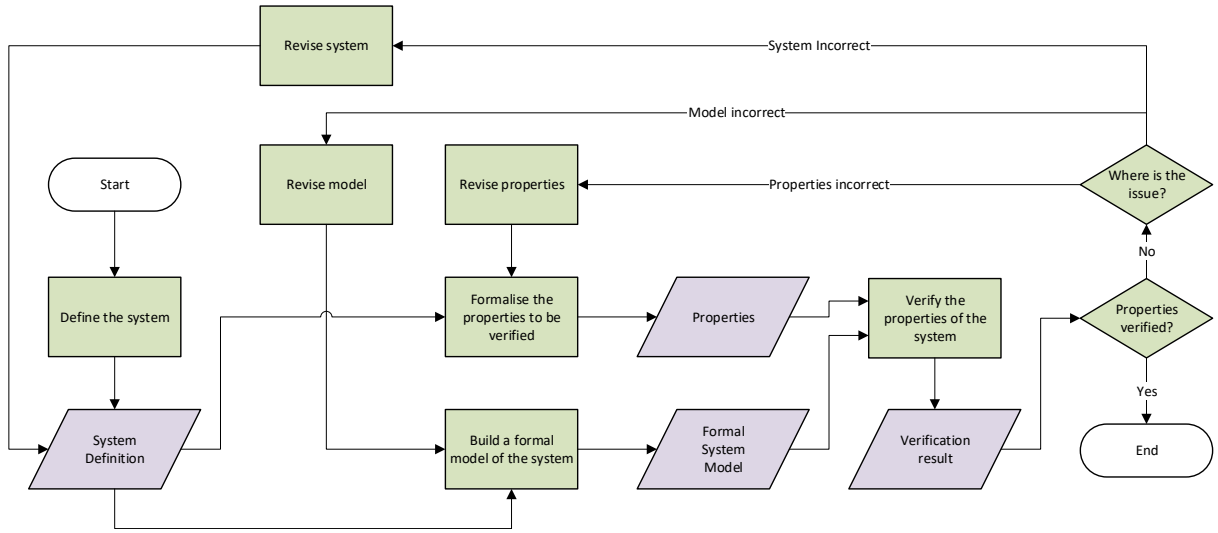

Figure 1. The formal verification methodology typically starts with a system definition from which formal models and properties to be verified are derived.

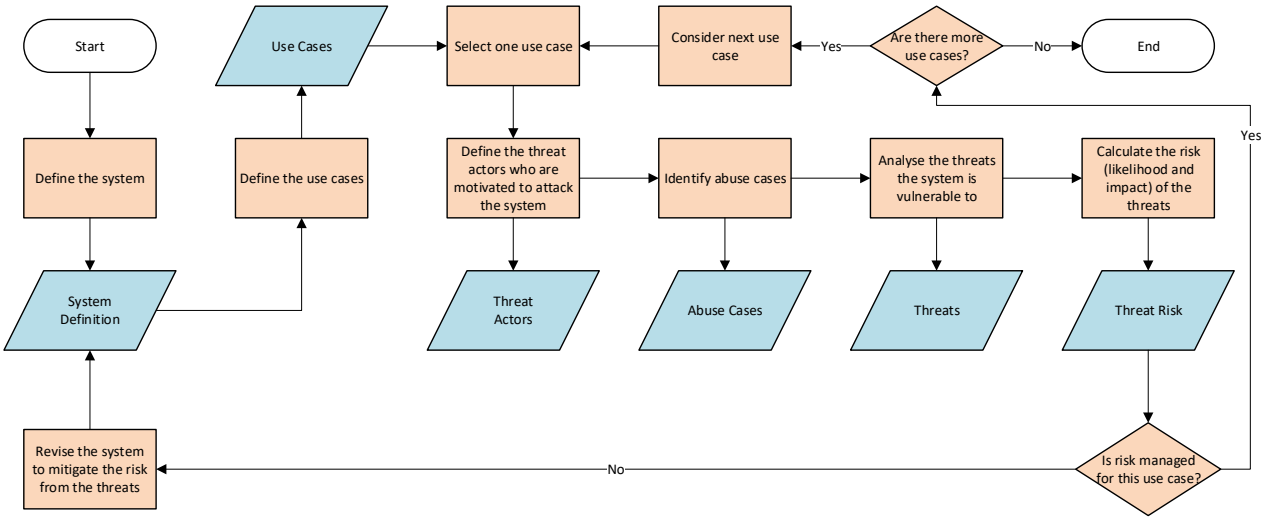

Figure 2. The threat modelling process begins with a system definition, from which use cases are derived and each is subsequently analysed from a cyber security perspective.

CTL-RP [49].

Alternatively, this could be conducted with a formal method such as Event-B [50]. Event-B, via its associated toolset, the Rodin Platform, facilitates verification using the Atelier B theorem prover. Its notation is ZF set theory and first-order logic. At this stage, the artefacts that are produced are a formal model (or a sequence of refined formal models) and the properties that we wish to verify against the model(s) that have been devised.

Next, we verify that the constructed model preserves the properties that we have defined. This can be achieved through model-checking with SPIN in the case of Promela, using a temporal logic theorem proving for systems specified in LTL or CTL, or theorem proving with Atelier B in the Rodin Platform for Event-B. If the properties are verified then we have completed our formal verification. Alternatively a probabilistic verification approach could be used, for example using the PRISM Probabilistic Model Checker [51]. In this case, the developer must set some threshold that indicates when the verification of a property is considered to be sufficient/acceptable.

However, if the properties could not be verified against the model then we must assess whether the system itself, our formal model or the properties contain errors. This may cause us to revise any of these and then progress again from the point that the change was made, as illustrated in Figure 1.

In addition to these formal methods, there are a variety of tools and techniques available for use in autonomous robotic systems with each offering benefits and limitations [16]. The methodology described in Figure 1 is not specific to any formal

method and we envision its use for whichever technique that the developer chooses.

Although informal verification techniques such as simulation-based testing and experiments with the real system should also be used to complement the formal system verification [9], our work here focuses only on formal verification. However, as part of our future work we plan to extend this methodology to encompass the results from these informal verification techniques.

Threat Modelling

Threat modelling is a technique commonly used for identifying security issues in software or systems [52]. Threat modelling involves identifying all possible threats to a system, irrespective of whether or not they can be exploited [53]. These threats are analysed in terms of the likelihood and harm of their exploitation, and the risk is then managed or accepted.

There are many tools and methodologies for performing threat identification and categorisation (e.g., STRIDE [54], PASTA [55] amongst others). An example of a high-level process for performing modelling threats is presented in Figure 2. Threat modelling can be performed at various stages of a system's development but it is ideally performed early and revised as changes are made to a system during development. For example, threat modelling could be performed on a system model before implementation to understand what threats the functionality will be subject to, it can be performed when aspects of the system change to understand the impact of those changes and it can be performed on a system implementation to understand the implementation specific threats.

To begin with, the system being investigated must be described. Unlike for formal verification, this description does not need to be a formal definition of the system, however, it must be sufficiently detailed to understand the functionality and interactions of the system. Once the system has been described, the use cases that describe how the system will be used can be defined. These use cases specify how the system will be used, by whom and what the user's goal is when interacting with the system.

For each use case, several steps are taken to evaluate how an adversary will be attacked and the risk associated with the attack. To begin with, the threat actors who are motivated to attack the system in this use case are identified. This involves specifying their goals, motivation, capabilities, resources and presence. The use cases can then form the basis of misuse or abuse cases [56, 57]. These abuse cases detail how the threat actors can interact with the system to achieve their goals. The threats to the system can now be considered from these abuse cases. Using the threat actors, abuse cases and identified threats, the likelihood and impact of these threats can be evaluated. These values can be summarised as the *risk* to the system. Finally, if the risk is not sufficiently low (allowing it to be accepted) then the risk must be managed. Risks can be managed through transference, avoidance or mitigation. The latter two of these methods require the system definition to be revised. If this managed risk results in a residual risk that is deemed to be sufficiently low, then the next use case should be analysed using the same procedure.

The majority of these tasks output information (as illustrated by the blue rhombuses in Figure 2) which will be used by subsequent tasks during the security analysis. Note that this information should not be discarded when performing an analysis of a subsequent use case, as many threat actors will have an interest in attacking different use cases to achieve their goals. Therefore, this information should be refined over the course of the analysis.

4. SECURITY-MINDED VERIFICATION METHODOLOGY

In this section, we describe our proposed security-minded verification methodology based on the methodologies for formal verification and threat modelling that were described in the previous section.

In general, formal verification and threat modelling are distinct processes which are usually performed independent of one another. Therefore, a naïve approach to integrating these processes might consist of first performing formal verification and then subsequently carrying out threat analysis. Both the threat analysis or formal verification may result in requiring modifications to the system, thereby forcing a new system verification or analysis of the security threats to the system. This process could be iterated until the system meets its requirements. However, this can be very ineffective since each independent analysis can lead to significant changes, and convergence can become a challenge.

We propose the security-minded verification methodology, illustrated in Figure 3, as a way to combine the usually distinct approaches to formal verification and threat modelling. Our approach to integrating security and verification follows a tightly coupled procedure and is inspired by techniques such as in the agile software development methodology. Agile development has been recognised to be well-suited to complex systems. A recent study by Ahimbisibwe et al. [58] examined 148 research papers through a systematic review to compare agile development to traditional methods. Of the agile method articles, 88.4% were addressing Technical Complexity as a critical success factor (CSF) compared to 39% of articles concerning traditional methods. The authors ranked 27 categories as CSFs from the literature based on number of occurrences. For Agile methods, Technological Uncertainty ranked first and Project Complexity ranked fourth (compared to 24^{th} and 25^{th}, respectively, for traditional methods).

To address these CSFs our security-minded verification methodology can be integrated into the system development in a similar manner as testing is performed in Continuous Integration. Where the security-minded verification methodology is first evaluated on an abstract system model, that is created before a system is implemented. Each time the system is modified, the system model is updated and the security-minded verification methodology is iterated upon until a successful verification is performed where the security risk is appropriately managed. This iterative, agile approach should can not only be managed at development time, but also in operation, similar to many other agile and dynamic approaches. Such an approach of performing continual, through-life adaptation and verification can be accomplished *procedurally* with ease. We do not introduce the operations element here, for the sake of simplicity in describing the approach and ease of representation through a flow-chart. It should be recognised, however, that continually refining a system in operation introduces a number of other challenges - monitoring, resolving and updating that may have an impact on operation and cost, so is non-trivial in practice.

Our approach is depicted in Figure 3; the top half corresponds to the formal verification methodology described in Figure 1

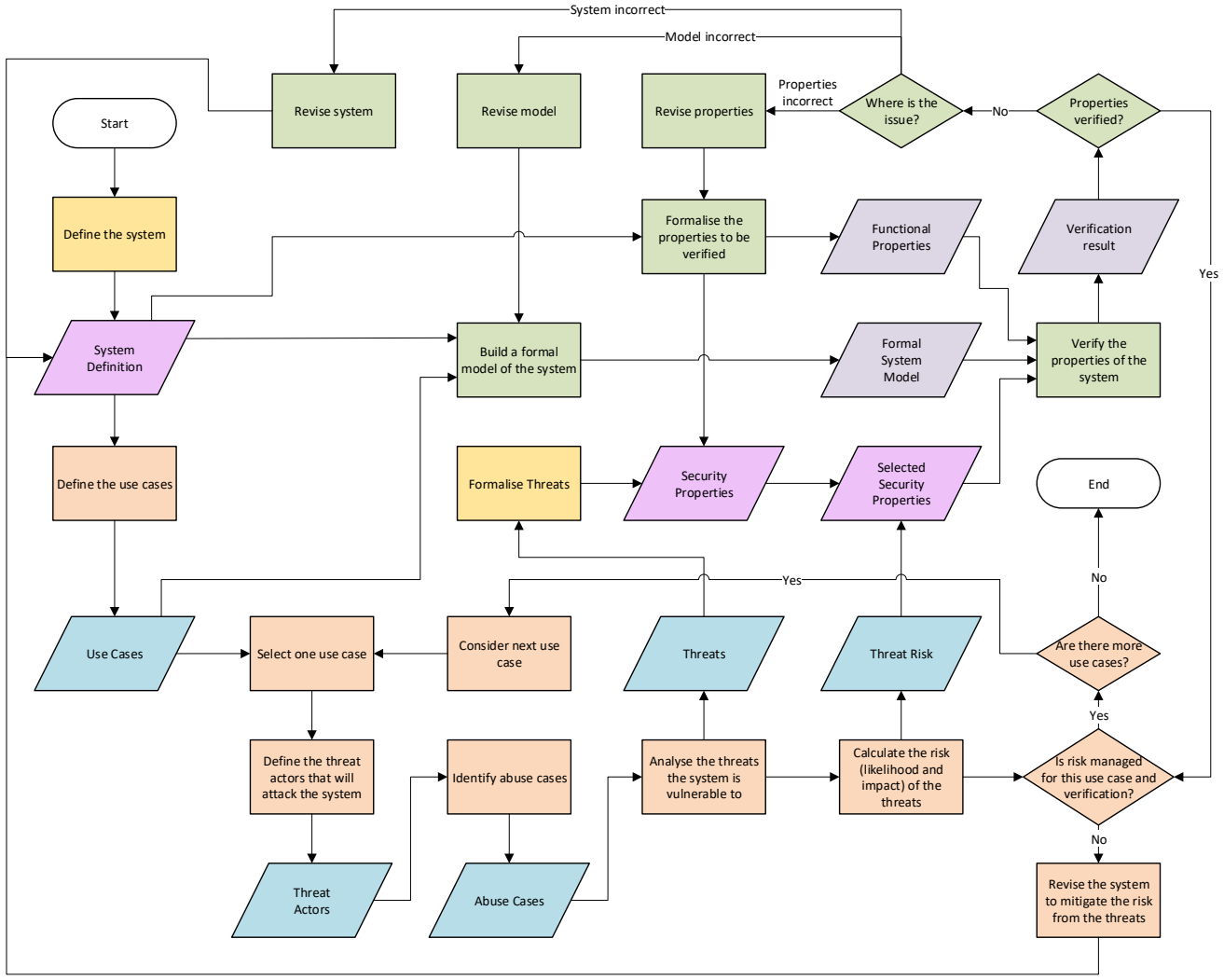

Figure 3. Integrated approach to combining verification and security analysis. Crucially, this security-minded verification approach allows us to use results from the threat modelling when both building a formal model of the system and formalising the properties to be verified.

and the bottom half to the threat modelling approach outlined in Figure 2. Instead of performing formal verification and a security analysis sequentially, aspects of these processes are performed in parallel. The fundamental change is that the security properties are formalised and checked as part of the verification of the system. Previously, these properties would have been identified due to system changes if the threat modelling identified that the risk of a threat to the system was too high. By always performing verification of security properties the verification of the system has an explicit consideration of the important security aspects that have been identified.

We observe that both Figures 1 and 2 begin with a System Definition and this provides a unified starting point for the security-minded verification methodology in Figure 3. We strengthen this starting point from a formal verification perspective by incorporating the use cases identified via threat modelling during the construction of the formal model of the system. Interestingly, the integration of the use cases that are identified by the cyber security approach helps to provide developers building a formal model of the system with more detail and focuses them on the relevant scenarios.

Central to our approach is our use of the results from the threat modelling to devise formal *Security Properties* via the *Formalise Threats* step of Figure 3. By assessing the *Threat Risk*, we then choose which of these security properties are the most interesting/important and these are subsequently formally verified. If the verification is successful then we check whether the risk is managed and proceed to examine any further use cases.

We believe that this integrated approach to security-minded verification provides a more streamlined development process and that the use of the security use cases in the development of the formal model is hugely beneficial from a verification perspective. Likewise the formalisation and subsequent verification of those security properties of interest gives more weight to the security analysis and helps to investigate whether any threat mitigation put in place works correctly.

We briefly illustrate how this methodology can be applied to an autonomous docking example in the next section.

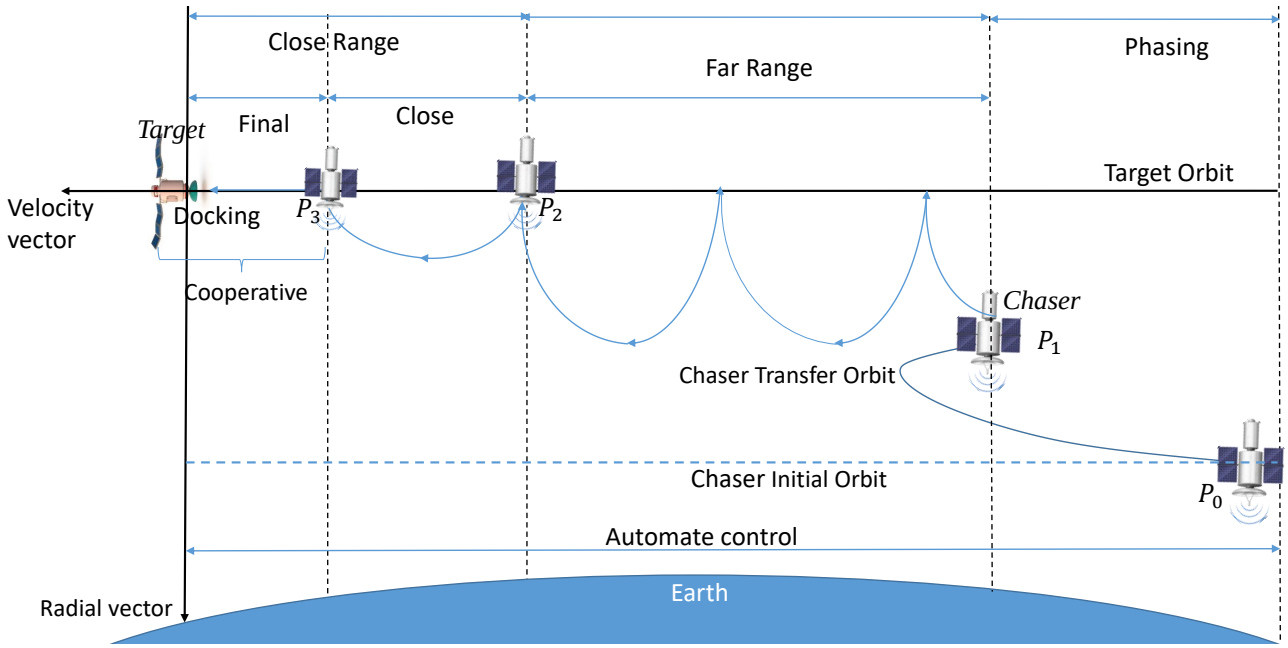

Figure 4. Autonomous on-orbit docking between a chaser and a target is comprised of multiple stages including far range and close range rendezvous.

5. Example: Autonomous Docking

In this section, we describe a simple example of autonomous docking and illustrate how our security-minded verification methodology (Section 4) can be applied in this setting.

There are multiple applications where autonomous docking may be useful in the future, such as automated orbital construction [59], orbital repair [60] and resource sharing between collaborating satellite swarms [61]. Automated docking will be facilitated by machine learning models controlling the actuation of thrusters and attitude control. Docking may be performed in a cooperative [62] manner that involves interaction such as via a communication protocol or guidance images [63] to allow the satellites to collaborate in order to dock.

Following the methodology described in Figure 3, we begin with a System Definition.

System Definition

Two satellites, a *target* and a *chaser* wish to dock to perform a resupply of the target. This docking is performed autonomously in a similar manner to the Orbital Express Demonstration System flight in 2007 [66]. A typical on-orbit servicing is mainly included: far range rendezvous, close range rendezvous and target capturing [67, 68, 69]. Figure 4 illustrates an autonomous on-orbit rendezvous. This consists of the following three basic operations.

Far range rendezvous: The main purpose of this operation (in Figure 4 from P_1 to P_2) is approaching the target orbit, reducing the approach velocity and synchronising the mission timeline.

Close range rendezvous: This operation is composed of two sub-phases: closing and final approach. The main purpose of the closing phase is to reduce the distance from the chaser to the target and achieve the appropriate position, velocity, attitude, etc. for the final approach sub-phase to begin. The methods of approach for proximity operations are: velocity vector [70] and radial vector [71]. The main purpose of the final approach is to satisfy the conditions of positions, velocities, relative attitude and angular rates of the docking. Normally, at the end of final approach (in Figure 4 at P_3), the chaser is about 250 m to 500 m to the target at radial vector. The chaser is located in a the corn-shaped corridor where the vertex is the target and the base faces to the chaser.

Docking: The docking starts when the target delivers the docking interfaces to the chaser. During this operation, the target cooperates with the chaser by adjusting its orbit (velocities and relative attitude) as required to complete the docking.

Next, we define the associated *Use Case* for this system.

Use Case

We consider the use case where the target cooperates with the chaser by maintaining its attitude so that the chaser can align docking ports with the target, accelerate towards the target and then decelerate before docking. This docking is performed autonomously by control systems locally on-board both satellites; there will be no human-in-the-loop impacting the operation of the docking.

At this point, Figure 3 shows that the process splits into two parallel activities. In particular, the combination of this use case and the system definition results in the development of a formal model of the system (we have not included the detailed formal model here). The other branch taken concurrently incorporates the threat modelling methodology described in Figure 2; we describe the constituent parts below.

Threat Actor	Example	Goals & Motivations	Capabilities	Environment	Resources
Privileged Insider	Employee	Financial gain; Discontent	High	Internal access with high permissions	Internal knowledge
Competitor	An organisation about to compete for a tender for services	Corporate espionage; Financial gain; Reputation damage	Organisation size related	Remote access	Organisation size related
Nation State	Geopolitical rival	State rivalry; Geopolitics	Sophisticated; Coordinated; Access to state secrets	Remote and internal access	Extensive knowledge; Extensive financial; Advanced equipment

Table 2. Threat actors for the autonomous docking based on [64] with the dimensions from [65]

Threat Actors

The space race began nearly six decades ago leading up to the launch of the first man-made satellite into orbit. Over time, an agreement for international collaboration in space has been made [72]. Currently, we are on the verge of a huge change in the space ecosystem due to the advancements in manufacturing, communication, and autonomous systems. The space ecosystem is no longer solely accessible to nation states but has also been opened to private organisations. In this regard, a race has emerged through the competition of private companies against each other, as well as state agencies. Such changes will impact the threat actors trying to attack space systems and bring new motivations, goals, capabilities, and resources. The attacks can now be performed by an individual or group of threat actors, organisations, or nations/states, driven by various motivations from personal satisfaction, ideology, financial gain to state rivalry [73]. A categorisation of threat actors is presented in NIST's Guide for Conducting Risk Assessments [64], and the corresponding dimensions have been presented in [65]. In our autonomous docking use case, the threat actors may potentially hinder or attack the system. A high-level view of the actors and their motivations is presented in Table 2.

Abuse-cases and Threats

An attacker may aim to compromise the target robot to disrupt the system. Ways in which the attacker may attempt to abuse the system include:

- Forcing the chaser to consume excessive, or indeed all, fuel during docking through the continuous or regular rotation of the target in such a way that docking cannot be completed.
- Forcing the chaser out of its intended region, perhaps even out of orbit, as it attempts to dock. This may be achieved by moving the target away from, rather than towards, the target.

The possible threats can be categorised using the STRIDE method. These categories are spoofing, tampering, reputation, information disclosure, denial of service, and elevation of privilege [54]. The target may be compromised with malicious firmware through an elevation of privilege (i.e. performing an unauthorised action) attack by one of the threat actors identified in Table 2. This will enable the attacker to have control of the target and realise the attacks described in the abuse cases above.

Formalise Threats/Security Properties

Based on the above threats, we can now identify and formalise the following security properties:

SEC1: Do not consume more than x kg of fuel while attempting to dock. This could be formalised as follows using the LTL operator $\Box$ (always now and in the future).

$$\Box(fuelremaining \geq x)$$

SEC2: Ensure that the position (or orbit) after successful docking or aborted docking is within $\pm y\%$ of the desired position (or orbit). This could be formalised as follows using the LTL operator $\bigcirc$ (in the next state).

$$\Box(dock \Rightarrow \bigcirc(orbit = initialorbit \pm y\%))$$

Risk Analysis

Risk is the possibility that a particular threat will cause an impact on system by exploiting a particular vulnerability. It is a function of likelihood and impact, and in its most basic form is given by *Risk = Likelihood x Impact* [31]. According to Kaplan and Garrick, risk analysis answers the following questions [74]:

- What can go wrong?
- What is the likelihood that it would go wrong?
- What are the consequences?

In the literature, there are currently numerous risk analysis and rating approaches in use. NASA has developed a framework for analysing the risks in their organisational procedures [75]. The framework has been divided into two stages as (i) Risk-Informed Decision Making and (ii) Continuous Risk Management. Performance shortfalls are analysed in terms of safety (e.g. avoidance of injury, fatality, or destruction of key assets), technical (e.g. thrust or output, amount of observational data acquired), cost (e.g. execution within allocated cost), and schedule (e.g. meeting milestones) [76].

Regarding the threats in the autonomous docking use case, we identify that the target satellite can be compromised by three threat actors as (i) Privileged insider, (ii) Competitor, and (iii) Nation state. These threat actors have different goals, capabilities, environment and resources as presented in Table 2. A nation state threat actor has more sophisticated capabilities and resources with internal or remote access to the target satellite. Therefore, we have quantified its attack likelihood as *High*. Besides this, a privileged insider may have high capabilities and resources but not as sophisticated as nation state threat actors, and the competitor's capabilities and resources are related to the size of the organisation. In order to differentiate the attack likelihoods of the privileged insider and competitor, their environment can provide an insight. For

example, a privileged insider has internal access whereas a competitor has remote access. Thus, we have quantified the likelihood of privilege insider as *Medium* and competitor as *Low*. It is important to recognise that these are example risk assessment levels for demonstrative purposes and a more formalised risk approach should be undertaken for specific scenarios.

Functional Properties

Next we consider the *Functional Properties* to be verified.

FUN1: The distance between the chaser and the target is always decreasing. This could be formalised as follows using LTL operators $\Box$ and $\bigcirc$.

$$\Box(distance(chaser, target) \geq \bigcirc distance(chaser, target))$$

FUN2: Once the appropriate conditions are met then they will dock successfully. This could be formalised as follows using LTL operators $\Box$ and $\Diamond$ (at some future moment).

$$\Box((cond_1 \wedge cond_2 \wedge \ldots \wedge cond_n) \Rightarrow \Diamond dock)$$

In particular, one of the $cond_i$ refers to the range allowed for velocity vector and radial vector as shown in Figure 4.

Verification of Properties

Next the properties should be verified on the formal model. In the case where the target is cooperative and maintains its attitude while the chaser attempts to dock, the properties (SEC1, SEC2, FUN1 and FUN2) should hold. However, in the case where the target is anti-cooperative and actively tries to avoid docking these properties no longer hold.

In particular, a violation of FUN1 may result in the violation of the fuel consumption security property (SEC1). Specifically, this might happen when an anti-cooperative target keeps moving away from the chaser to avoid docking. Similarly, FUN2 might not hold even when each of the $cond_i$ are met if the target exhibits this anti-cooperative behaviour.

Revise Properties/Model/System

In the presence of an anti-cooperative target, it is clear that mitigations are required so that the system becomes both safe and secure. One potential mitigation would be to add a description of the anti-cooperative behaviour so that the chaser can identify when the target is misbehaving and abort the mission rather than run out of fuel. However, this would mean that functional properties may no longer be able to be satisfied. The left hand side of the implication in FUN2 could be strengthened to incorporate the assumption that the target is co-operative. Similarly FUN1 could be re-written as an implication including a pre-condition about the target being co-operative, or not moving or similar.

Once this mitigation has been put in place then our methodology (shown in Figure 3) dictates that we must reevaluate the security risks of the system and potentially formalise new functional properties which are then verified. For example, the functional properties could change from "the target and chaser eventually dock", to "the target and chaser eventually dock if possible". The process continues until we are satisfied that there are no further threats to be considered and that the risk has been managed. If the system needs to be evaluated under a different threat model then the way in which the threat actors and abuse cases are defined can be updated and the process repeated. Changing the threat model may require system implementation changes, changes to functional and security properties, and will impact the system risk. Making changes to manage this new threat model will need to consider the previous threat models too and ensure those threats are sufficiently managed.

6. DISCUSSION

The space ecosystem is evolving rapidly with enhancements in communications technology, processing capability and process improvements. It is benefitting from developments such as the efficacy and affordability of artificial intelligence and the expansion of the Internet of Things. As such the market is creating new opportunities and attracting new entrants. This is not without challenges, as new entrants are unlikely to have the skills, experience, nor perhaps motivation to ensure that systems are secure and verifiable. Given the increasing importance of, and dependence on, these systems, addressing this challenge is becoming critical. Through the use of a tightly integrated and rapid development methodology, we can address both security and verifiability concerns most efficiently.

7. CONCLUSION AND FUTURE WORK

In this paper, we have presented a novel methodology for security-minded verification. We have combined the usually distinct methodologies for threat modelling and formal verification. The approach is intended to offer robust security and verification in a manner that maximises efficiency by ensuring that iterations are short and allow quicker convergence on a solution that meets requirements. We have illustrated the approach using the real example of autonomous docking scenario in space, that is currently being explored by the FAIR-SPACE Hub.

For future work, we will develop the verification and the threat modelling further. For example, we will develop detailed formal models on which to carry out verification of the identified properties and mitigations to the identified threats (both safety and security). We also intend to consider other space use cases, for example, concerning coordination, control and autonomy for planetary rovers.

In this work we have presented a methodology for performing security-minded verification of space systems, where we have focused solely on formal verification and have not included informal verification approaches such as testing and simulation. Whilst formal verification provides a proof of correctness of the software system, considering all paths through the system, usually it is only applied to an abstraction of the system and not to its full implementation. Simulation based testing and real world experiments can be applied to the implementation itself but only a subset of all program traces can be considered. The number of real world experiments may be limited due to the time they take or because of dangerous or difficult to access environments (such as space). More runs can be carried out via simulation based testing but again the simulation is an abstraction of the real world and only a subset of all program traces can be checked. Similarly the type of properties relating to these types of verification are different requiring formal semantics for the properties for formal verification but potentially incorporating softer, informal aspects for real world experiments. To achieve high levels of verification in a system, the formal and informal approaches to verification should be used in a complementary

fashion.

An example of the combination of formal verification, simulation based testing and real user experiments for a robot assistant scenario was discussed in [9]. The authors used the output from different verification methods is used to enhance the inputs to other methods to improve our confidence in the system. In the space domain or in other extreme environments real experiments in situ are unlikely to be possible but some aspects may be able to be investigated in test environments (such as terrestrial testbeds). Our future work will investigate including these aspects of informal verification in the presented methodology, as well as demonstrating them in representative use cases.

ACKNOWLEDGMENTS

The authors would like to thank Daniel Fowler for assisting with proof reading this paper.

This work is supported by grant EP/R026092 (FAIR-SPACE Hub) through UKRI under the Industry Strategic Challenge Fund (ISCF) for Robotics and AI Hubs in Extreme and Hazardous Environments.

REFERENCES

[1] W. W. Wu, E. F. Miller, W. L. Pritchard, and R. L. Pickholtz, "Mobile satellite communications," *Proceedings of the IEEE*, vol. 82, no. 9, pp. 1431–1448, 1994.

[2] C. Dong, J. Yang, W. Zhang, Z. Yang, N. Lu, J. Shi, P. Zhang, Y. Liu, and B. Cai, "An overview of a new chinese weather satellite fy-3a," *Bulletin of the American Meteorological Society*, vol. 90, no. 10, pp. 1531–1544, 2009.

[3] Y. Yang, J. Li, J. Xu, J. Tang, H. Guo, and H. He, "Contribution of the compass satellite navigation system to global PNT users," *Chinese Science Bulletin*, vol. 56, no. 26, p. 2813, 2011.

[4] L. F. Zarzalejo, L. Ramirez, and J. Polo, "Artificial intelligence techniques applied to hourly global irradiance estimation from satellite-derived cloud index," *Energy*, vol. 30, no. 9, pp. 1685–1697, 2005.

[5] I. F. Akyildiz and A. Kak, "The internet of space things/cubesats: A ubiquitous cyber-physical system for the connected world," *Computer Networks*, vol. 150, pp. 134–149, 2019.

[6] A. Poddey, T. Brade, J. E. Stellet, and W. Branz, "On the validation of complex systems operating in open contexts," *arXiv preprint arXiv:1902.10517*, 2019.

[7] J. Bhadra, M. S. Abadir, S. Ray, and L.-C. Wang, "A survey of hybrid techniques for functional verification," *IEEE Design and Test of Computers*, vol. 24, no. 02, pp. 112–122, 2007.

[8] J. Bhadra, M. S. Abadir, L.-C. Wang, and S. Ray, "A survey of hybrid techniques for functional verification," *IEEE Design & Test of Computers*, no. 2, pp. 112–122, 2007.

[9] M. Webster, D. Western, D. Araiza-Illan, C. Dixon, K. Eder, M. Fisher, and A. G. Pipe, "A Corroborative Approach to Verification and Validation of Human–Robot Teams," in *International Journal of Robotics Research*, 2019, To Appear.

[10] A. Cornell, "Five key turning points in the american space industry in the past 20 years: Structure, innovation, and globalization shifts in the space sector," *Acta Astronautica*, vol. 69, no. 11-12, pp. 1123–1131, 2011.

[11] K. Drost, "Successful market entry in the european commercial space industry," *MSc Thesis, TU Delft*, 2017.

[12] O. F. E. Cooperation and Development, *The Space Economy at a Glance 2014*. OECD Publishing, 2014.

[13] D. Livingstone and P. Lewis, "Space, the final frontier for cybersecurity?" *Chatham House, September*, 2016.

[14] J. James, "Adaptive control for post-dock maneuvers with an unknown semi-cooperative object," in *IEEE Aerospace Conference*. IEEE, 2016, pp. 1–10.

[15] M. Farrell, M. Bradbury, M. Fisher, L. A. Dennis, C. Dixon, H. Yuan, and C. Maple, "Using threat analysis techniques to guide formal verification: A case study of cooperative awareness messages," in *International Conference on Software Engineering and Formal Methods*, ser. LNCS, vol. 11724. Springer, 2019, pp. 471–490.

[16] M. Luckcuck, M. Farrell, L. A. Dennis, C. Dixon, and M. Fisher, "Formal specification and verification of autonomous robotic systems: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, p. 100, 2019.

[17] M. Farrell, M. Luckcuck, and M. Fisher, "Robotics and integrated formal methods: Necessity meets opportunity," in *Integrated Formal Methods*, ser. LNCS, vol. 11023. Springer, 2018, pp. 161–171.

[18] G. Brat, E. Denney, D. Giannakopoulou, J. Frank, and A. Jónsson, "Verification of autonomous systems for space applications," in *IEEE Aerospace Conference*. IEEE, 2006, pp. 11–pp.

[19] G. Brat, D. Drusinsky, D. Giannakopoulou, A. Goldberg, K. Havelund, M. Lowry, C. Pasareanu, A. Venet, W. Visser, and R. Washington, "Experimental evaluation of verification and validation tools on martian rover software," *Formal Methods in System Design*, vol. 25, no. 2-3, pp. 167–198, 2004.

[20] M. C. B. Alves, D. Drusinsky, J. B. Michael, and M.-T. Shing, "Formal validation and verification of space flight software using statechart-assertions and runtime execution monitoring," in *International Conference on System of Systems Engineering*. IEEE, 2011, pp. 155–160.

[21] K. Y. Rozier and J. Schumann, "R2u2 in space: System and software health management for small satellites," in *9th Annual Workshop on Spacecraft Flight Software (FSW-2016)*, 2016.

[22] J. Schumann, P. Moosbrugger, and K. Y. Rozier, "R2u2: monitoring and diagnosis of security threats for unmanned aerial systems," in *Runtime Verification*, ser. LNCS, vol. 9333. Springer, 2015, pp. 233–249.

[23] A. Tarasyuk, I. Pereverzeva, E. Troubitsyna, T. Latvala, and L. Nummila, "Formal development and assessment of a reconfigurable on-board satellite system," in *International Conference on Computer Safety, Reliability, and Security*, ser. LNCS, vol. 7612. Springer, 2012, pp. 210–222.

[24] N. Tabarcia and D. Stroescu, "Space–an area of strategic importance to nato," *Strategic Impact, 2/2010*, vol. 2, pp. 19–29, 2010.

[25] M. R. Maheshwarappa and C. P. Bridges, "Software defined radios for small satellites," in *2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. IEEE, 2014, pp. 172–179.

[26] D. Housen-Couriel, "Cybersecurity and anti-satellite capabilities (asat) new threats and new legal responses," *Journal of Law & Cyber Warfare*, vol. 4, no. 3, pp. 116–149, 2015.

[27] A. Drozhzhin, "Russian-speaking cyber spies from turla apt group exploit satellites," Sep 2015. [Online]. Available: https://www.kaspersky.co.uk/blog/turla-apt-

exploiting-satellites/6210/
[28] B. Unal, "Cybersecurity of nato's space-based strategic assets," *Chatham House, July*, 2019.
[29] G. Falco, "Cybersecurity principles for space systems," *Journal of Aerospace Information Systems*, vol. 16, no. 2, pp. 61–70, 2018.
[30] T. Harrison, K. Johnson, and T. G. Roberts, *Space Threat Assessment 2018*. Center for Strategic & International Studies, 2018.
[31] CCSDS, "Security Threats against Space Missions," The Consultative Committee for Space Data Systems (CCSDS), Informational Report, Dec. 2015, CCSDS 350.0-G-3. [Online]. Available: https://public.ccsds.org/Pubs/350x1g2.pdf
[32] A. Waller and R. Craddock, "Managing runtime re-engineering of a system-of-systems for cyber security," in *2011 6th International Conference on System of Systems Engineering*. IEEE, 2011, pp. 13–18.
[33] C. Steward Jr, L. A. Wahsheh, A. Ahmad, J. M. Graham, C. V. Hinds, A. T. Williams, and S. J. DeLoatch, "Software security: The dangerous afterthought," in *International Conference on Information Technology-New Generations*. IEEE, 2012, pp. 815–818.
[34] C. Snook, T. S. Hoang, and M. Butler, "Analysing security protocols using refinement in iUML-B," in *NASA Formal Methods Symposium*, ser. LNCS, vol. 10227. Springer, 2017, pp. 84–98.
[35] J. Whitefield, L. Chen, F. Kargl, A. Paverd, S. Schneider, H. Treharne, and S. Wesemeyer, "Formal analysis of v2x revocation protocols," in *Security and Trust Management*, ser. LNCS, vol. 10547. Springer, 2017, pp. 147–163.
[36] G. Vanspauwen and B. Jacobs, "Verifying protocol implementations by augmenting existing cryptographic libraries with specifications," in *Software Engineering and Formal Methods*, ser. LNCS, vol. 9276. Springer, 2015, pp. 53–68.
[37] L. Huang and E.-Y. Kang, "Formal verification of safety & security related timing constraints for a cooperative automotive system," in *Fundamental Approaches to Software Engineering*, ser. LNCS, vol. 11424. Springer, 2019, pp. 210–227.
[38] S. Schneider and R. Delicata, "Verifying security protocols: An application of csp," in *Communicating Sequential Processes. The First 25 Years*, ser. LNCS. Springer, 2005, vol. 3525, pp. 243–263.
[39] S. Schneider, "Formal analysis of a non-repudiation protocol," in *Computer Security Foundations Workshop*. IEEE, 1998, pp. 54–65.
[40] F. Aarts, J. De Ruiter, and E. Poll, "Formal models of bank cards for free," in *International Conference on Software Testing, Verification and Validation Workshops*. IEEE, 2013, pp. 461–468.
[41] J. De Ruiter and E. Poll, "Formal analysis of the EMV protocol suite," in *Joint Workshop on Theory of Security and Applications*, ser. LNCS, vol. 6993. Springer, 2011, pp. 113–129.
[42] E. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. MIT Press, 2000.
[43] M. Fisher, *An Introduction to Practical Formal Methods Using Temporal Logic*. Wiley, 2011.
[44] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi, "The Temporal Analysis of Fairness," in *Proceedings of the Seventh ACM Symposium on the Principles of Programming Languages*, January 1980, pp. 163–173.
[45] G. J. Holzmann, *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley, 2003.
[46] E. M. Clarke and E. A. Emerson, "Design and Synthesis of Synchronisation Skeletons Using Branching Time Temporal Logic," in *Proceedings of the Workshop on the Logic of Programs*, ser. LNCS, D. Kozen, Ed., vol. 131. Springer, 1981, pp. 52–71.
[47] M. Fisher, C. Dixon, and M. Peim, "Clausal Temporal Resolution," *Transactions on Computational Logic*, vol. 2, no. 1, pp. 12–56, Jan. 2001.
[48] U. Hustadt and B. Konev, "TRP++ 2.0: A temporal resolution prover," in *Automated Deduction—CADE-19*, ser. LNAI, vol. 2741. Springer, 2003, pp. 274–278.
[49] L. Zhang, U. Hustadt, and C. Dixon, "A Resolution Calculus for the Branching-Time Temporal Logic CTL," *ACM Transactions on Computational Logic*, vol. 15, no. 1, pp. 1529–3785, 2014.
[50] J.-R. Abrial, *Modeling in Event-B: system and software engineering*. Cambridge University Press, 2010.
[51] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: A Tool for Automatic Verification of Probabilistic Systems," in *Proceedings 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. LNCS, vol. 3920. Springer, 2006, pp. 441–444.
[52] A. Shostack, *Threat modeling: Designing for security*. John Wiley & Sons, 2014.
[53] S. Myagmar, A. J. Lee, and W. Yurcik, "Threat modeling as a basis for security requirements," in *Symposium on requirements engineering for information security*, vol. 2005. Citeseer, 2005, pp. 1–8.
[54] A. Shostack, "Experiences threat modeling at microsoft." in *MODSEC@ MoDELS*, 2008.
[55] T. UcedaVelez and M. M. Morana, *Risk Centric Threat Modeling: process for attack simulation and threat analysis*. John Wiley & Sons, 2015.
[56] J. McDermott and C. Fox, "Using abuse case models for security requirements analysis," in *Proceedings 15th Annual Computer Security Applications Conference*. IEEE, 1999, pp. 55–64.
[57] P. Hope, G. McGraw, and A. I. Antón, "Misuse and abuse cases: Getting past the positive," *IEEE Security & Privacy*, vol. 2, no. 3, pp. 90–92, 2004.
[58] A. Ahimbisibwe, R. Y. Cavana, and U. Daellenbach, "A contingency fit model of critical success factors for software development projects: A comparison of agile and traditional plan-based methodologies," *Journal of Enterprise Information Management*, vol. 28, no. 1, pp. 7–33, 2015.
[59] C. J. Stein and M. Reiher, "Automated selection of active orbital spaces," *Journal of chemical theory and computation*, vol. 12, no. 4, pp. 1760–1771, 2016.
[60] G. Hirzinger, B. Brunner, R. Lampariello, K. Landzettel, J. Schott, and B.-M. Steinmetz, "Advances in orbital robotics," in *Proceedings IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 898–907.
[61] G. Bonnet and C. Tessier, "Collaboration among a satellite swarm," in *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. ACM, 2007, p. 54.
[62] M. Ciarcia, A. Grompone, and M. Romano, "A near-optimal guidance for cooperative docking maneuvers," *Acta Astronautica*, vol. 102, pp. 367–377, 2014.
[63] C. Pirat, F. Ankersen, R. Walker, and V. Gass, "Vision based navigation for autonomous cooperative docking of cubesats," *Acta Astronautica*, vol. 146, pp. 418 – 434, 2018.
[64] R. S. Ross, "Guide for Conducting Risk Assessments," National Institute of Standards and Technology, Tech. Rep., Sep. 2012, SP 800-30 Rev. 1.

[65] Q. Do, B. Martini, and K.-K. R. Choo, "The role of the adversary model in applied security research," *Computers & Security*, vol. 81, pp. 156–181, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404818306369
[66] A. Ogilvie, J. Allport, M. Hannah, and J. Lymer, "Autonomous Satellite Servicing Using the Orbital Express Demonstration Manipulator System," in *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2008. [Online]. Available: http://robotics.estec.esa.int/i-SAIRAS/isairas2008/Proceedings/SESSION%2014/m113-Ogilvie.pdf
[67] W. Xu, B. Liang, C. Li, Y. Liu, and X. Wang, "A modelling and simulation system of space robot for capturing non-cooperative target," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 15, no. 4, pp. 371–393, 2009.
[68] D. A. Barnhart, R. Rughani, J. J. Allam, B. Weeden, F. A. Slane, and I. Christensen, "Using Historical Practices to Develop Safety Standards for Cooperative On-Orbit Rendezvous and Proximity Operations," in *69th International Astronautical Congress (IAC)*, Bremen, Germany, Oct. 2018, Paper Code: IAC-18,D1,5,8,x45161. [Online]. Available: https://iafastro.directory/iac/paper/id/45161/summary/
[69] W. Xu, B. Liang, C. Li, and Y. Xu, "Autonomous rendezvous and robotic capturing of non-cooperative target in space," *Robotica*, vol. 28, no. 5, p. 705–718, 2010.
[70] Y.-Z. Luo, G.-J. Tang, and Y.-J. Lei, "Optimal multi-objective linearized impulsive rendezvous," *Journal of guidance, control, and dynamics*, vol. 30, no. 2, pp. 383–389, 2007.
[71] K. Yamanaka, K. Yokota, K. Yamada, H. Koyama, K. Tsukahara, S. Yoshikawa, and T. Nakamura, "Guidance and navigation system design of r-bar approach for rendezvous and docking," in *17th AIAA International communications satellite systems conference and exhibit*, 1998, p. 1299.
[72] A. Cracknell and C. Varotsos, "Fifty years after the first artificial satellite: From sputnik 1 to envisat," *International Journal of Remote Sensing*, vol. 28, pp. 2071–2072, 2007.
[73] J.-M. Bockel, *The Future of the Space Industry*. NATO Parliamentary Assembly - Economic and Security Committee (ESC), 2018.
[74] S. Kaplan and B. J. Garrick, "On the quantitative definition of risk," *Risk analysis*, vol. 1, no. 1, pp. 11–27, 1981.
[75] *Agency Risk Management Procedural Requirements*, NASA, 2016, NASA NID 8000-108, NPR 8000.4A.
[76] H. Dezfuli, A. Benjamin, C. Everett, G. Maggio, M. Stamatelatos, R. Youngblood, S. Guarro, P. Rutledge, J. Sherrard, C. Smith *et al.*, *NASA Risk Management Handbook, Version 1.0*, 2011.

BIOGRAPHY

Carsten Maple *is Professor of Cyber Systems Engineering in WMG at the University of Warwick, where he is the Director of Research in Cyber Security. Carsten has an international research reputation having published over 200 peer-reviewed papers and his research has attracted millions of pounds in funding and has been widely reported through the media. He is Principal Investigator (PI) at the EPSRC/GCHQ Academic Centre of Excellence in Cyber Security Research, leads various projects on the security of CAVs and is a fellow of the Alan Turing Institute and member of the ENISA CARSEC Expert Group.*

Matthew Bradbury *received his MEng and PhD degrees in computer science from the Department of Computer Science at the University of Warwick, Coventry, UK in 2013 and 2018 respectively. Since 2018 he has been a Research Fellow in WMG, University of Warwick, Coventry, UK. His research interests include security and privacy aspects of Internet of Things, including wireless sensor networks, intelligent transportation systems and space systems. He received the best-in-session award at InfoCom 2017.*

Hu Yuan *is a research fellow in the University of Warwick, where his research focus on the security and privacy aspects of IoT, including internet of bio-nano things, vehicular communication networks, user behaviours identification and further space system. He received his PhD in wireless communications from University of Warwick, UK in 2016, and MSc. in Communications Engineering from University of York, UK in 2012.*

Marie Farrell *is a Research Associate in the Department of Computer Science at the University of Liverpool working primarily on the EPSRC funded FAIR-SPACE Hub but also participating in the RAIN and ORCA Hubs. She received her PhD from Maynooth University, Ireland in 2017. Her current area of research is in using and combining formal methods to reason about and provide certification evidence for robotic systems that are to be deployed in hazardous environments.*

Clare Dixon *is a Reader in the Department of Computer Science, University of Liverpool, Liverpool, U.K. Her research interests include formal verification and temporal and modal theorem-proving techniques, in particular applied to robotics and autonomous systems. She is a member of the Autonomy and Verification Laboratory and the Centre for Autonomous Systems Technology at the University*

of Liverpool. She is also a member of the British Standards Institution AMT/10 Committee on Robotics.

Michael Fisher *holds a Royal Academy of Engineering Chair in Emerging Technologies in the Department of Computer Science at the University of Liverpool. He is the Director of the Multi-Disciplinary Centre for Autonomous Systems Technology, University of Liverpool, Liverpool, U.K. His current research interests include formal verification for the certification, safety, ethics, and reliability of autonomous systems. He is a member of both the British Standards Institution AMT/10 Committee on Robotics and the IEEE P7009 Standard for Fail-Safe Design of Autonomous Systems. He is on the editorial boards of Applied Logic and Annals of Mathematics and Artificial Intelligence journals. He is a Corner Editor of the Journal of Logic and Computation.*

Ugur Ilker Atmaca *received his BSc in electronic and communication engineering from Suleyman Demirel University, Turkey, in 2013. After working in industry, he received his MSc in computer science from the University of Reading, UK, in 2017. He is is currently pursuing the PhD degree at the Warwick Manufacturing Group, the University of Warwick, UK. His research interests include security and privacy in intelligent transportation systems.*