Towards Security Minded Verification

A Case Study of Cooperative Awareness Messages

Matthew Bradbury

Cyber Security Centre, WMG, University of Warwick

6th November 2019

Introduction

1. Summary of the FAIR-SPACE Hub
2. Aims of Security Minded Verification
3. Case study of Cooperative Awareness Messages
4. Discussion on encountered issues

The Team

WMG
THE UNIVERSITY OF WARWICK


Carsten Maple


Hu Yuan

UNIVERSITY OF LIVERPOOL


Michael Fisher


Clare Dixon


Louise Dennis

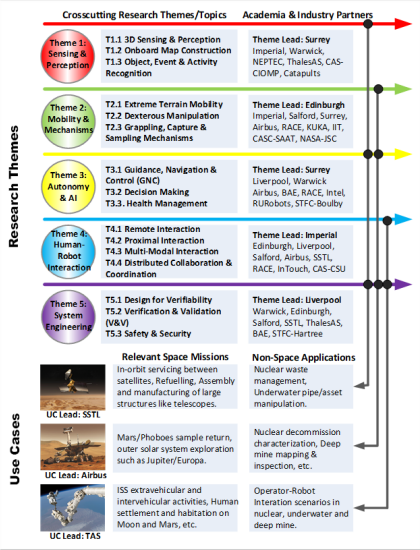

Marie Farrell

FAIR-SPACE Hub I

The aim of the Future AI and Robotics for Space (FAIR-SPACE) Hub is to **go beyond the state-of-the-art** in robotic sensing and perception, mobility and manipulation, on-board and on-ground autonomous capabilities, and human-robot interaction, and to **enable long-lived robotic operations in space**.

- To merge the best available off-the-shelf hardware/software solutions with trail-blazing innovations, new standards and frameworks, leading to a constellation of **space RAI prototypes and tools**.
- To accelerate the prototyping of autonomous systems in a scalable way, where the innovations and methodologies developed can be **rapidly adopted by the space industry**.

FAIR-SPACE Hub II



- Five cross-cutting research themes underpinning major industry-led challenges.
- Each research theme addresses a set of scientific topics and objectives through collaborative projects between academia & industry.
- Outputs from selected themes are coherently combined within industry-defined use cases to demonstrate new knowledge and technologies.

FAIR-SPACE Hub III

Use-Cases:

Orbital: robots for repairing satellites, assembling large space telescopes, manufacturing in space, removal of space junk

Planetary: for surveying, observation, extraction of resources, and deploying infrastructure for human arrival and habitation

Human-robot: will target astronauts-robot interoperability aboard the International Space Station or for the future Moon Village.

How do space systems relate to vehicular systems?

Both are:

- Complicated systems
- Resource constrained (computation, fuel, energy, and others)
- Safety and security critical

Future similarities:

- Include new kinds of connectivity and autonomy
- CAVs expected to operate in more hazardous environments in future (e.g., high radiation environments)

But why start with vehicular systems? $\rightarrow$ We know more about them and we want to transfer our knowledge of them to space systems.

Problem Statement

"Manufacturers providing vehicles, and other organisations supplying parts for trials will need to **ensure** that all vehicle systems have appropriate security measures to manage data security and the risk of unauthorised data access."

— §2.17, Code of Practice: Automated vehicle trialling, CCAV, Feb 2019

How do we ensure security of these systems?

Security Minded Verification

Verification and Validation (V&V)

- Verification: Does the system meet the specification?
- Validation: Does the system meet the needs of the user?

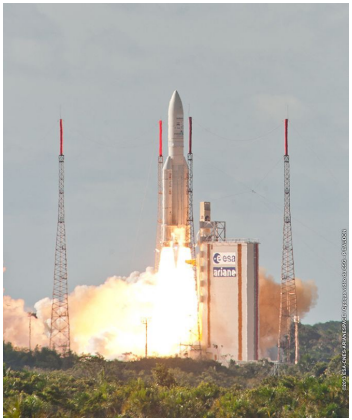
In other words:

- Verification: Are you building the thing right?
- Validation: Are you building the right thing?

— Barry Boehm, Software Engineering Economics, 1981

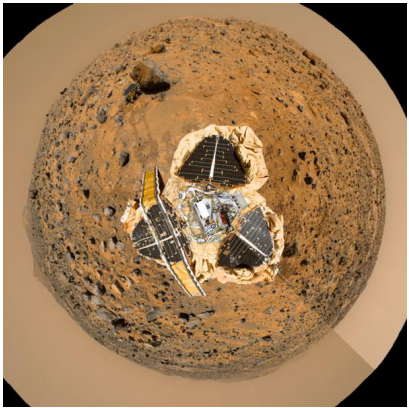
Verification could be informal (testing/simulation) or formal (proofs).

Ariane 5 Flight 501 (04/06/1996)



30 seconds after launch the rocket deviated 90 degrees from flight path. Destroyed by aerodynamic stresses. Caused by overflow due to casting a 64-bit float to a 16-bit integer. Backups failed for the same reason.

Mars Pathfinder (04/07/1997)



(a) © NASA/JPL

Priority inversion in scheduling system led to system resets. A program was deployed that allowed the problem to be fixed.

Miller and Valasek (2014)

(a) © WIRED

Remote compromise of a vehicle that allowed messages to be injected onto the CAN bus[1]. Remote physical control of certain aspects of the vehicle was made possible.

Insight into the system was possible by allowing custom firmware to be flashed to the Uconnect system (WiFi, Cellular, Bluetooth and CAN).

[1] `http://illmatics.com/Remote%20Car%20Hacking.pdf`

Remove Keyless Entry Thefts (Garcia et al. 2016)

Inexpensive SDRs allow easy eavesdropping, recording and replying to messages. Insecure (P)RKE schemes can allow vehicles to be unlocked and stolen[2].

[2] https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_garcia.pdf

Formal Verification Considerations

- These systems are large and complex — Where should efforts be focused? What properties are of interest?
- How can we formalise properties?
- Can not prove everything (usually).
- Machine learning is especially challenging.

Formal Verification Properties

- **Liveness**: The system eventually does something.
- **Safety**: The system does not do anything bad.

Example: Two lane road into single lane bridge

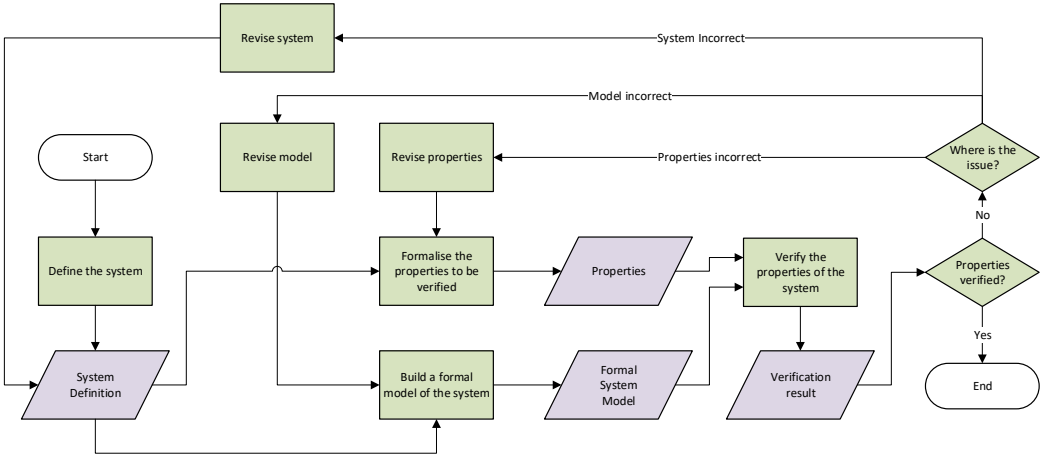
- **Liveness**:
  If a traffic light is red it will eventually become green.
- **Safety**:
  When one traffic light is green the other is red.

Consider additional properties too:

- **Fairness**: Over a long period of time,
  both traffic lights are green for similar periods of time.

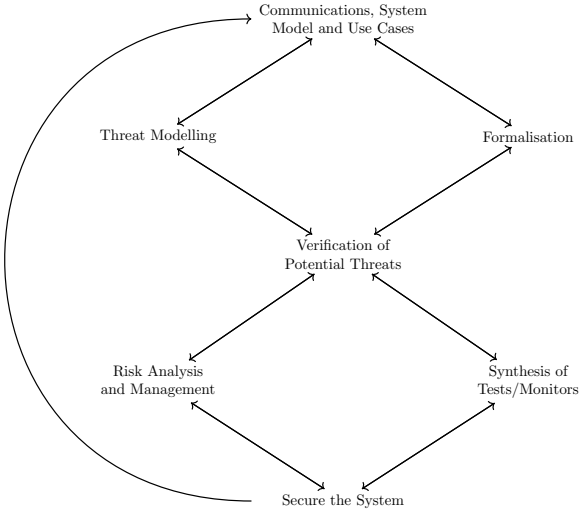
Figure: Railway bridge, Ballycarry station — August 2018 © Albert Bridge

How to Perform Formal Verification?

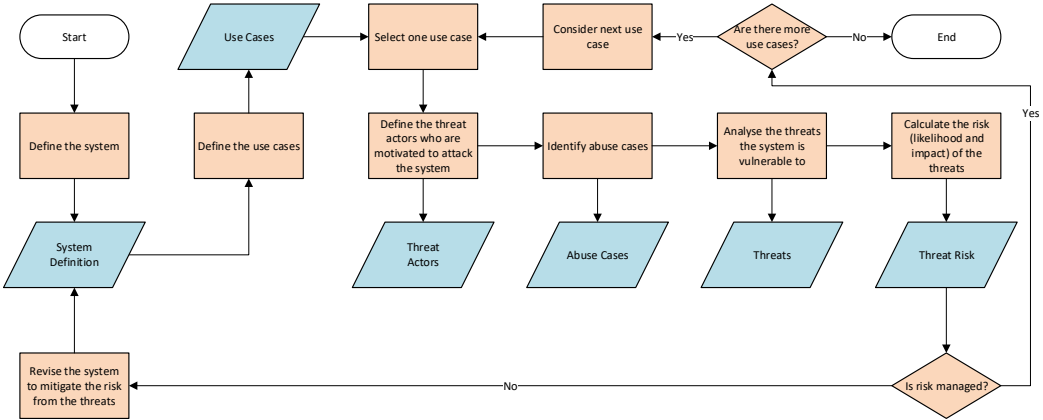


Security Minded Verification

- Which aspects of the system are going to be important to protect?
- How do we identify these areas?
- What if we are dealing with systems of system?
- Which security properties are important and how do we formalise them?



How to Perform Security Analysis and Revision?



Requirements for Security Analysis

- To understand how to attack a system we need to know:
  - Who will perform the attack,
  - What their capabilities are,
  - Why they are attacking the system,
  - Where their attacks originate from,
  - How their attacks reach the target
- We have to focus on specific aspects of the system (the two use cases) to constrain our analysis to a reasonable size
- But we need a high-level overview of the rest of the ecosystem to understand how the use cases can be attacked

Adversaries

Threat Actor	Motivation	Resources	Access	Knowledge
Nation-States	State rivalry, Geopolitical	Extensive	Remote coordinated	Easy to develop knowledge
Commercial Competitors	Corporate espionage, Financial gain	Company size-related	Remote coordinated	High to moderate knowledge
Organised Crime	Financial gain	Moderate availability	Remote coordinated	Moderate knowledge
Amateur Cracker	Personal satisfaction	Minimal	Remote	Limited
Insiders	Discontent, Financial	Minimal	Privileged access	Internal Knowledge

Adversary Presence

Presence of the adversary is specified relative to the target of the attack

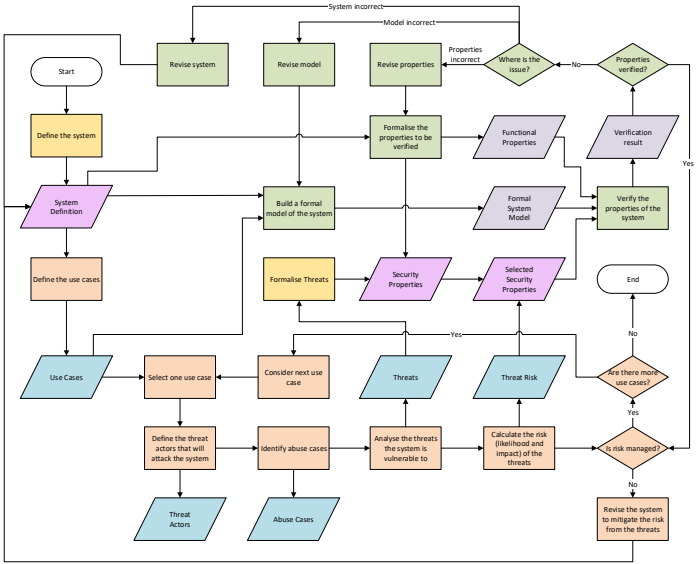
- **Internal**: Has physical access to the internals of the target
- **Local**: Adversary has physical access to the external surface of the target
- **Semi-local**: Adversary is physically nearby the target (e.g., within direct communication or sensor range)
- **Remote**: Adversary is physically far from the target (e.g., access via the internet or satellite network)

The presence impacts the kinds of threats an adversary can carry out.

Threat Modelling

- Many different approaches to modelling threats.
- Want to think about "what can go wrong in the system?"
- Not all possible threats will be applicable, depending on the adversary, its capabilities and its presence.
- How are threats classified (CIA, STRIDE, or others)?

Security Minded Verification



A Case Study of Cooperative Awareness Messages

Case Study Introduction

- Vehicles in Connected and Autonomous Vehicle (CAV) Systems communicate over a wireless network (e.g., IEEE 802.11p / cellular).
- Ensuring that both cyber security and safety issues are addressed during software development is crucial.
- This case study employs informal threat analysis techniques to guide a verification of the Cooperative Awareness Message (CAM) protocol.
- Two distinct verification tools were used to verify the CAM protocol at different levels of abstraction (SPIN at system-level and Dafny at algorithmic-level).

Cooperative Awareness Messages (CAMs) — ETSI EN 302 637-2

- CAMs are heartbeat messages that are broadcasted from vehicles.
- CAMs include information such as generation time, velocity, position, heading and others. Aim is to provide context information to other vehicles for safety purposes.
- CA Basic Service layer responsible for two services:
  1. sending of CAMs including their generation and transmission
  2. receiving of CAMs and the modification of the receiving vehicles' state as a result
- CAMs are sent unencrypted to eliminate possibilities of overheads this would impose.
- CAMs are sent with a digital signature in order to authenticate the sender.

Specialising STRIDE for CAM

Spoofing: attacker sends messages masquerading as another vehicle.

Tampering: attacker tampers with a message sent by another vehicle.

Repudiation: a vehicle can deny sending a message that it has actually sent.

Information Disclosure: vehicles only receive messages intended for them.

Denial of Service: messages are not sent within a reasonable time frame.

Elevation of Privilege: attacker can obtain ability to perform actions by a more privileged group.

Considering the Threats: T, I, E

- Tampering is prevented via digital signatures (providing integrity checks).
- CAMs are intended for all who receive them so there is no need to consider Information Disclosure properties.
- ITS stations have certificates which indicate if they are allowed to send CAMs (either in general or for a specific role), CAMs are not processed by other ITS stations if the CAM is not consistent with the permissions in the certificate.
- The most relevant/important threats are Spoofing, Denial of Service and Repudiation.

Considering the Threats: S, D, R

Spoofing: a vehicle pretending to be another vehicle and sending false information could potentially cause vehicles to collide.

Denial of Service: If a vehicle sends too many CAMs then the network becomes overloaded. If CAMs are not sent frequently enough then insufficient context information about vehicles is available.

Repudiation: A compromised vehicle could claim to not have sent a CAM when in fact it has sent one.

Model-Checking with Promela/SPIN: Basic Scenario

Autonomous speed change to avoid collisions. Leader and tail vehicles broadcasting CAMs.

Actions by middle vehicle:

- If no CAMs are received then vehicles continue with speed unchanged.
- If it receives exactly one CAM then sets its own speed to half the speed in the CAM.[3]
- If it receives two CAMs then it sets its own speed to be the average of the two speeds (rounded down).

[3] This only occurs at initialisation when the speed of the other vehicle is 0.

Model-Checking with Promela/SPIN: Default Conditions

Default Conditions:

- the leader chooses a random discrete speed 10, 20, 30, 40, 50, 60 or 70 at each time step
- the tail similarly chooses a random discrete speed at each time step
- we ran the system for 100 time steps with a round-robin interleaving concurrency between vehicles

Model-Checking with Promela/SPIN: Verified Property

Safety property: the speed of the middle vehicle is never *much* different (difference of more than 51) to the speed of the leader or of the tail.

We write this in temporal logic as:

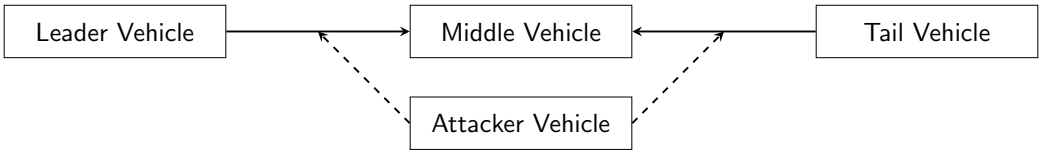
$$\Box(\mathit{big\_speed\_difference} \Rightarrow \bigcirc \neg \mathit{big\_speed\_difference})$$

Due to how SPIN works this needs to be rewritten as a check that is never true:

$$\Diamond(\mathit{big\_speed\_difference} \wedge \bigcirc \mathit{big\_speed\_difference})$$

Where $\Box$ means *always*, $\Diamond$ means *eventually* and $\bigcirc$ means *in the next state*.

Investigating Spoofing



Investigating Spoofing

```
0  proctype attacker(chan l_in, t_in){   /* attacker */
1    printf("attacker: starting\n");
2    bool head = 0;
3    bool tl = 0;
4    A:    (clock > 10); /* wait until under way */
5      if
6       :: (head = 0) -> printf("attacker: inserting vspeed of 10\n");
7            l_in!10; l_in!10; head = 1; goto A;
8                  ...
9       :: (head = 0) -> printf("attacker: inserting vspeed of 70\n");
10           l_in!70; l_in!70; head = 1; goto A;
11      :: (tl = 0) -> printf("attacker: inserting tspeed of 10\n");
12           t_in!10; t_in!10; tl = 1; goto A;
13                 ...
14      :: (tl = 0) -> printf("attacker: inserting tspeed of 70\n");
15           t_in!70; t_in!70; tl = 1; goto A;
16      :: (clock ≤ 100) -> goto A;
17      :: (clock > 100) -> goto FIN;
18     fi;
19   FIN:  printf("attacker: finishing\n")
20 }
```

Investigating Spoofing

- Safety property no longer holds with the attacker present.
- Spoofing attack impacts on the safety of the system.
- This simple example scales up to more complex versions of Spoofing attacks.

Deductive Verification with Dafny

- Two basic methods: `sendCAM` and `receiveCAM`.
- Focus on Denial of Service and Repudiation security threats at algorithmic level.
- Use the following simplified structure of CAMs:
  `CAM(id:int, time:int, heading:int, speed:int, position:int)`

Sending CAMs

```
0 method sendCAM(T_CheckCamGen: int, T_GenCam_DCC: int)
1 returns (msgs: seq<CAM>, now: int)
2 requires 0 < T_CheckCamGen ≤ T_GenCamMin;
3 requires T_GenCamMin ≤ T_GenCam_DCC ≤ T_GenCamMax;
4 ensures T_GenCam_DCC * |msgs| ≤ now ≤ T_GenCamMax * |msgs|;
5 ensures |msgs| ≥ 2 ⟹ ∀ i: int • 1 ≤ i < |msgs| ⟹
6     T_GenCam_DCC ≤ (msgs[i].time - msgs[i-1].time) ≤ T_GenCamMax;
7 ensures |msgs| = MaxMsgs;
```

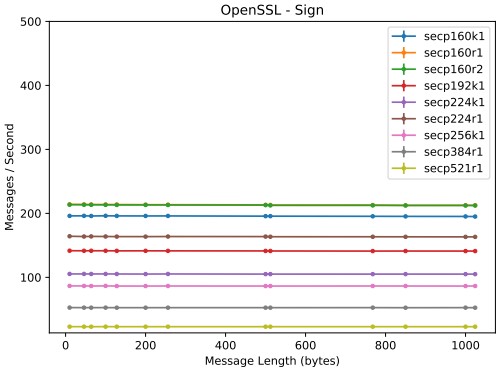
- Denial of Service: messages are sent on time and arrive within specified time bounds.
- Dafny does not support real-time systems so we had to manually keep track of time.
- Corresponding loop invariants support the postconditions above.

Receiving CAMs

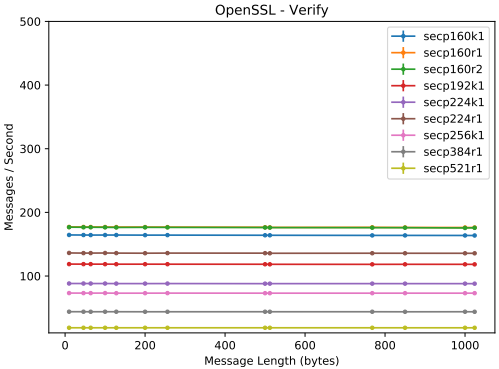
```
0 method receiveCAM(fromid: int, cams: seq<CAM>, now: int) returns (brake: bool)
1 requires 0 ≤ fromid < |cams|;
2 requires fromid = cams[fromid].id;
3 ensures !(now - cams[fromid].time > T_GenCamMax)
4         ∧ Sign(Magnitude(cams[fromid].heading)) = Sign(Magnitude(GetHeading(now)))
5         ∧ GetSpeed(now) - cams[fromid].speed < 0 ⟹ brake;
6 ensures now - cams[fromid].time > T_GenCamMax ⟹ !brake;
```

- Non-repudiation: we require that the received CAM came from a vehicle with a valid id and that the vehicle claiming to send the CAM did actually send one.
- The documentation requires that CAMs cannot be forwarded to other vehicles and our preconditions capture this.
- We also capture a safety property related to when the vehicle should brake.
- If the security property is violated and an attacker sends a false message to the receiving vehicle then there could be a collision.

Sign/Verify Limitations



(a) Sign



(b) Verify

Conclusions

- Identifying and formalising security properties is hard when the security properties are not functional aspects of a system.
- Assumptions need to be made about dependencies. For example, we need to rely on other work to prove security properties of ECDSA sign and verify.
- In this use case we needed an application to verify that made use of CAMs, otherwise we lacked safety and liveness properties.
- A more detailed system model is needed to investigate DoS provisions. It is likely that formal verification may not be suitable to analyse how DoS is protected against.

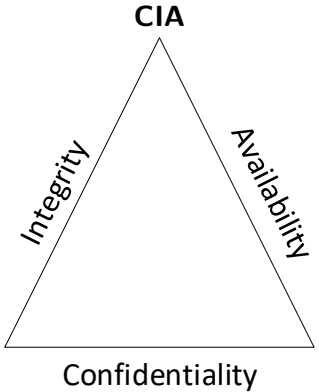
Summary

- We have presented a case study that uses cyber security threat analysis techniques to guide formal methods practitioners in verifying security properties.
- Modelled the system at different levels of abstraction to investigate and verify properties related to STRIDE threats.
- Formal methods are used to focus security analysis on to specific areas highlighted by an informal cyber security analysis.
- Future work: define a more general methodology for combining threat analysis techniques and formal methods.
- We intend to apply this methodology to space systems, e.g. CAM messages between rover platoons on the Moon/Mars.

Applying Lessons Learnt from Connected Autonomous Vehicles to Space Systems

Is Encryption Always Needed? I

It depends on what security properties you want.



STRIDE

- Authenticity
- Integrity
- Non-repudiation
- Confidentiality
- Availability
- Authorisation

Is Encryption Always Needed? II

- **Authenticity** — Need a way to prove the identity of the sender of the message.
- **Integrity** — Need a way to validate the received contents is the same as the sent contents.
- **Non-repudiation** — Need a way to prove the identity of the sender of the message.
- **Confidentiality** — Encryption.
- Availability — Protocol and operation considerations.
- Authorisation — Need a way to prove the identity of the sender of the message and check what permissions they have.

Is Encryption Always Needed? III

Security Property	Cryptographic Primitive			
	Hash	MAC	Digital Signature	Encryption
Integrity	✓	✓	✓	✗
Authenticity	✗	✓	✓	✗
Non-repudiation	✗	✗	✓	✗
Confidentiality	✗	✗	✗	✓

MAC cannot provide non-repudiation as at least two entities have the secret key.

Thank you for listening.

Any questions?