Towards autonomous cyber-defence: using co-operative decision making for cybersecurity*

Madeline Cheah[1,*], Jack Stone[1], Samuel Bailey[1], Peter Haubrick[1], David Rimmer[1], Matt Lacey[1] and Mark Dorn[1]

[1]*Cambridge Consultants, 29 Cambridge Science Park, Milton Road, Cambridge, CB0 5DW*

Abstract

Fully autonomous decision-making for cyber-defence (the ability to make expert-level defensive choices without human intervention) is desirable but challenging. This is particularly so for operational technology because of its cyber-physical nature and the need to take into account multiple dimensions of context. Our contribution is the creation and substantial extension of our co-operative decision-making framework for cyber-defence (Co-Decyber). This framework allows us to break up a large multi-contextual action space into smaller decisions for multiple agents to optimise between. We have applied this framework to a vehicle platooning scenario (the linking of two or more trucks in a convoy) . This paper discusses development since our last published work, which is based on increased complexity by defending against a more sophisticated attack (diversion of the convoy using GPS message spoofing) using more agents. Results show that Co-Decyber agents are able to successfully defend against an attack and recover the situation. We conclude that this framework is viable and once mature, will assist in fully autonomous cyber-defence of operational technology.

Keywords

autonomous, cyber-defence, cybersecurity, automotive, artificial intelligence, multi-agent, reinforcement learning, autonomous cyber-defence, operational technology

1. Introduction

Autonomous decision making is highly desirable when defending systems from cyber-attack. This is even more so in operational technology where there are time constraints (due to safety implications; for example, if the response is too slow and a vehicle crashes as a result) along with limitations in the operational environment such as intermittent connectivity. However, it is also challenging. Operational technology is heterogeneous, may be made up of legacy systems and protocols, and can be opaque due to its bespoke, proprietary, or confidential nature.

Cybersecurity for operational technology also has to overcome context-awareness issues. This includes:

- The rare signals problem – for example, an emergency brake signal is always an anomaly, but may not always be a malicious action, and a cyber-defence system should be able to differentiate between these two situations.
- User awareness and response – unlike in enterprise contexts, the first line of defence is usually an operator with no cybersecurity expertise (e.g., the driver of the vehicle), whose focus is not on cybersecurity, and who may also be overloaded in operational contexts, therefore leading to unexpected interactions with the defence mechanisms.
- Safety interactions - whereby security operations may unduly impede or interfere with safety mechanisms that could lead to danger from other contexts (for example by causing a denial of service through filtering out engine control messages).

CAMLIS'24: Conference on Applied Machine Learning for Information Security, October 24–25, 2024, Arlington, VA
*Corresponding author.
madeline.cheah@cambridgeconsultants.com (M. Cheah)
https://www.linkedin.com/in/madelinecheah (M. Cheah)

Our contribution in this paper is a substantial extension of our co-operative decision-making framework for cyber-defence (which we call Co-Decyber). To the best of our knowledge, this is the first such framework to be applied to autonomous cyber-defence (ACD) of vehicles.

We apply this framework to operational technology, namely an automated vehicle platoon as proof-of-concept. We extend our previous work [1] in the following ways:

- Created a significantly more complex attack scenario (GPS divert) which requires cooperation between different agents to resolve the cyber-attack. To support this, we have extended our simulator with new NATO Generic Vehicle Architecture (NGVA) NGVA modules (see Section 3.2), creating models for modules that do not yet exist in the NGVA specification (such as the vehicle-to-vehicle or V2V module), and created three variants of the scenario (stateful, stateless and varied attack source – see Section 3.3) to test against our updated Co-Decyber framework.
- Trained existing agents to work with our additional modules (GPS0 and GPS1) as well as an entirely new kind of agent (NAV_RESET) to defend against the GPS divert attack.
- Created a model evaluation framework which allows us to assess performance of different agents systematically.

The rest of this paper is structured as follows: we explore related work in Section 2 before expanding on the Co-Decyber framework in Section 3, including the modelling techniques and scenarios we considered. We then cover our training approach in Section 4 before presenting our results in Section 5. Discussions of our work is given in Section 6 and conclusions and future directions outlined in Section 7.

2. Related work

Cyber-threats are growing more sophisticated, driven by the acceleration in pace of technological change. The threat landscape is also evolving quickly, with more systems becoming connected, more complex and more intelligent. This is propelling research in autonomous and automated cyber-defence, in both offensive and defensive security, covering red-teaming, intrusion detection and prevention, incident response and security planning [2].

ACD can be defined as a decision-making system with competent capabilities that should be capable of detecting an attack and responding by recovering, proving or correcting the vulnerability in real time without human intervention [1]. A discussion of definitions for ACD (and its components such as agents) may be found in [3].

There are broadly two categories in terms of ACD frameworks: rule-based and machine-learning based.

2.1. Rule-based frameworks

The most mature rule-based methods can be found in incident response playbooks, security incident and event management (SIEM) and security orchestration, automation and response (SOAR) products [4]. However, these are focused on enterprise defence and are typically dependent on a centralised architecture. They also require computational resource and guaranteed connectivity that is not typically found in operational situations. There is also academic work looking to extend these frameworks, especially in the challenges of mission or context-awareness. For example, there has been work looking at automating attack-defence trees using a game theoretic approach, by modelling the optimal benefits for attackers or defenders to help decide on the best response [5].

This work is supported by studies that deal with explicit expressions and proofs of equivalence, such as that between game theory and attack-defence trees [6]. Some use graphical models, e.g., [7], which have been precomputed to estimate possible attack paths of an ongoing attack to mitigate in real-time. Other approaches include automating the risk assessment process (again using attack-defence trees) for deciding on optimal security controls [8].

These methods are all high-confidence methods, with plentiful and rigorous mathematical and algorithmic foundations and tools for cyber-attack and defence expression. However, there are limitations with such approaches, including high knowledge barriers required for modelling systems and interactions, a barrier that is compounded by the fact that operational technology is most likely to be legacy, bespoke or proprietary. Further limitations arise from the rigidity of expression and increased abstraction from the real-world which could also lead to missing out on real-world edge cases. Furthermore, rule-based systems can be limited in their adaptability when facing novel attacks or dynamic situations [9], which can lead to a lack of resilience.

2.2. Multi-agent approaches

The primary alternative that the security community has explored to address such limitations is through the use of probabilistic approaches, including machine learning. Machine learning - and in particular reinforcement learning (RL) - has been the most widely explored due to the ability to take into account context. Uses include red teaming (such as simulating spoofing activity or malware) and defences, including detection of deception and preventing data injection. A comprehensive survey of reinforcement learning in ACD may be found in [10].

Multi-agent approaches are more recent. Agents within a multi-agent reinforcement learning (MARL) framework can be competitive, collaborative or both. There are also two broad schools of research: the first is in training MARL agents using game theoretic approaches in adversarial settings [11, 12] using intelligent red agents to train against. The second is through optimisation of defences through use of the RL exploration versus exploitation paradigm.

There has been a strong growth in interest in using MARL, with many agreeing that intelligent autonomous agents will be present in future resilience, defence and operational technology contexts [13] and that cooperation may be more successful than single or multiple independent agents [14]. Uses explored have included generating defence strategies [15], intrusion detection [16] and learning tactical responses (from scratch) in IT networks [17].

All approaches have similar challenges, in that optimisation is not universal and is highly dependent on the definitions of the defensive or tactical actions taken by the MARL cyber-defence, and that these definitions require input from human subject matter experts. Additionally, the environment is non-stationery and there are sparse reward challenges (discussed further in Section 3.4). Many of the approaches described remain theoretical and have rarely been transferred to real world implementation.

3. The Co-Decyber framework

Our approach is based on cooperative MARL, where agents attached to a unit of functionality work together to protect the system globally (see Section 3.4 and 3.5). In this section, we describe the underlying attack and defence paradigm that allows us to scope and define our cyber-defence work in Section 3.1. We expand on our high-level scenario (automated vehicle platooning) in Section 3.2 before describing our attack scenario (GPS divert) in Section 3.3. We discuss mission level impacts and the reward structure we use in Section 3.4 before giving an overview of the entire Co-Decyber framework in Section 3.5.

3.1. Attack and defence modelling

We use attack-defence tree methodology [18] to map out attacks and defences (Figure 1).

Each defence (I, II, III etc.) can be considered an action that Co-Decyber can take. The set of defences is therefore Co-Decyber's action space (although due to training constraints, we have not implemented a complete coverage of this defensive action space). Note also that since Co-Decyber is intended to provide operational cyber-defence, design-time defences are out of scope in our current research.

We perform no computation on the tree; the attack-defence tree paradigm was chosen to help scope our action space and communicate conceptually how we break down the decision making to give rise

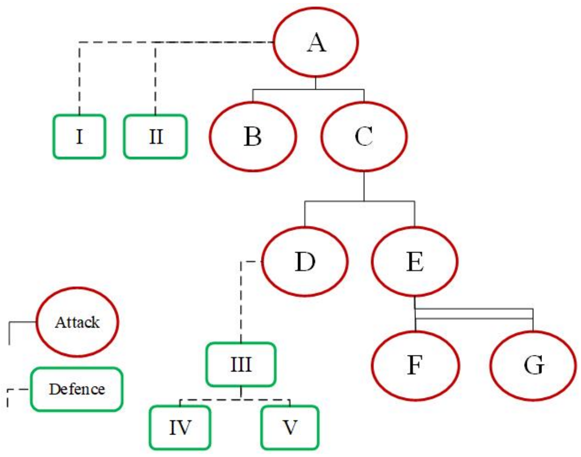

Figure 1: Fig. 1. Generic attack-defence tree. Defence I or II can be considered defences to attack A. Fulfilment of defence IV or V are considered to fulfil the requirements of defence III, which is a defence to attack D (a sub-attack of attack C). Double line notation between F and G is for an AND gate; all others are OR gates.

```
ATTACK GOAL: A
      (-OR: I)
      (-OR: II)
      -OR: B
      -OR: C
           -OR: D
                (-OR:III)
                     (-OR:IV)
                     (-OR:V)
                -OR: E
                     -AND: F
                     -AND: G
```

Figure 2: Adapted attack-defence tree notation: the visual diagram in Figure 1 can be represented by the notation above.

to the multi-agent architecture.

The advantages and disadvantages of the multi-agent architecture in the context of our framework are discussed in more detail in Section 3.5. For ease of reading, we use an adapted notation (see Figure 2) to represent all attack-defence trees in this paper.

As an example of how we translate between attack-defence tree and multi-agent architecture, we provide an attack of using a false fire alarm (see Figure 3) to delay the convoy (this is the scenario we used in an earlier study [1] to establish the Co-Decyber framework).

```
ATTACK GOAL: DELAY CONVOY
      -OR: Message alteration
      -OR: Unauthorised publication
           -AND: Hijack publisher
           -AND: Send false fire alarm
                (-OR: Mitigate attack)
                     (-OR: Isolate using ACL)
                     (-OR: Block alarm)
                          (-OR: 1 alarm)
                          (-OR: All alarms)
                     (-OR: Power down alarm src)
```

Figure 3: Attack-defence tree modelling false fire alarm.

To enable Co-Decyber's decision making, each action is assigned to individual but interconnected RL agent (see Figure 4). These agents (aggregated into a node) negotiate between themselves on what the best course of action should be. Each node is coupled with a unit of functionality to be defended, to ensure that defensive actions are appropriate and relevant to that particular piece or type of functionality (see Section 3.5). There is also a hierarchy of agents:

- *Lower-level agents* are responsible for atomic actions (e.g. to isolate or block)
- *Higher level aggregation agents* are responsible for co-ordinating these actions, including deciding between these actions, based on bottom-up input from lower-level agents. The lower-level agents essentially act as a feature extractor for the aggregation agents.

An illustrative Co-Decyber node can be seen in Figure 4.

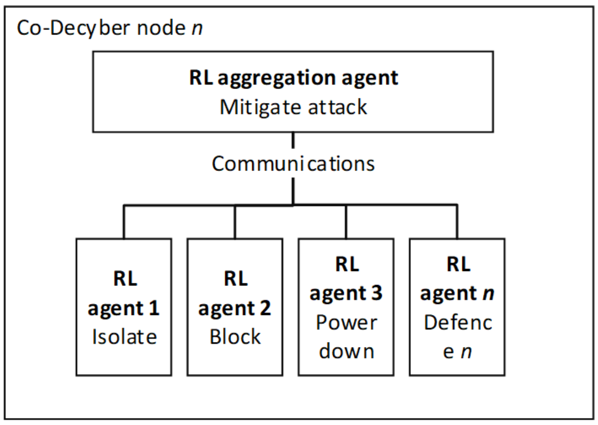

Figure 4: A Co-Decyber node; note that the RL agent defences are consistent with the defences in the attack-defence tree in Figure 3

We traded off the additional complexity of a hierarchical architecture against the disadvantages of a flat peer-to-peer architecture. The latter would have meant that:

- Information from other agents will have been delayed by one time tick. This means that the other agents may have already taken “bad” actions because they did not have full context when they made their own decision.
- Individual agents must know about each other. Many of the advantages of a composable multi-agent approach would be lost as agents must be trained as part of a specific collection.

We posit that the hierarchical approach gives the best of both worlds. Aggregation agents are more tightly coupled with the specific modules they are trained for to determine the “best” action, but the lower-level modules have a simpler development approach and can be re-used or re-composed into different nodes for different modules. Hierarchical agents therefore provide module specific knowledge and arbitrate between different low-level agents without requiring retraining of the low-level agents.

Some nodes are simple (containing one RL agent), and others are complex (containing multiple RL agents).

3.2. Platooning vehicles

The Co-Decyber framework has been developed using a high-level leader-follower scenario (Figure 5), in which military vehicles are aiming to move cargo from a rear operating base to a forward base. The leader vehicle is manned, and the follower vehicle has an automated system that follows the leader’s path.

We chose this scenario for the richness of the attack-defence space and its relevance to generation-after-next technologies as well as to defence. High level attacks and defences in the scenario are

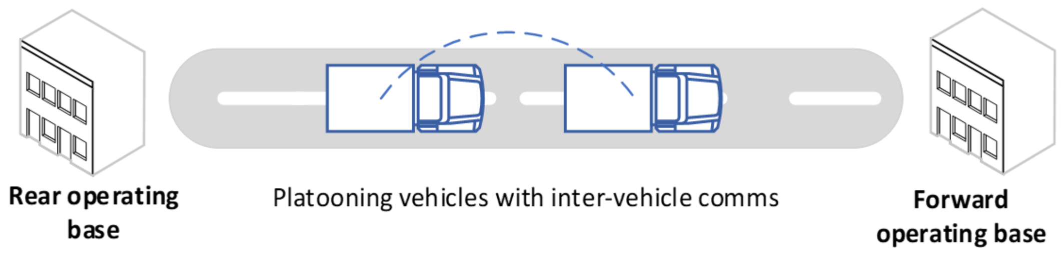

Figure 5: Autonomous vehicle platooning scenario

modelled using attack-defence tree methodology to scope the attack and explore the associated defence action space (see Section 3.1).

To enable training and generate the datasets we require, we built a training simulator (Figure ??) to represent a NATO Generic Vehicle Architecture compliant system to be defended. There are two components to the simulator: the environment and agent inference. These components allow exploration of different environments and agent configurations, and are extensible to take into account future scenarios we may want to generate datasets for. These are joined using an OpenAI Gym [19] interface, with multi-agent support coming from PettingZoo [20].

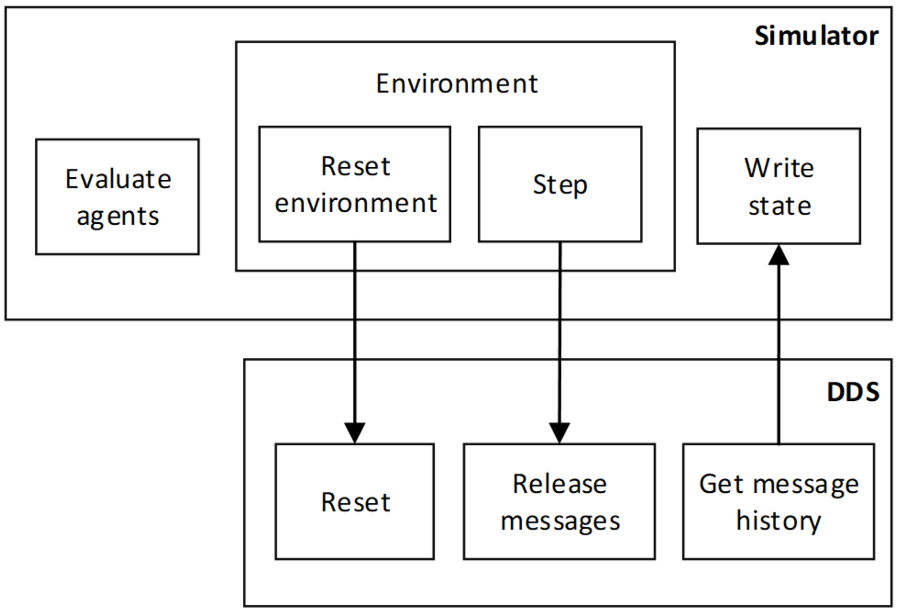

Figure 6: High-level simulator architecture

NGVA is a framework that introduces pluggable modules for reconfigurability (called GVA modules), connected together using a Data Distribution Service (DDS) backbone. DDS is a publish-subscribe system which provides shared topics on which all modules can read and/or write. A GVA module describes a specific unit of functionality e.g. engine controls, sensor systems, navigation. We use NGVA to inform our attack-defence trees as well as the Co-Decyber architecture.

For this paper, we have extended our simulator to include simulation of more GVA modules (in addition to the HMI, MEDIA, ECU and SWITCH modules which we had previously modelled [1]). This includes:

- NAV, which refers to the navigation reference module which traditionally integrates position information from GPS and inertial measurements. The NAV module contains two low-level

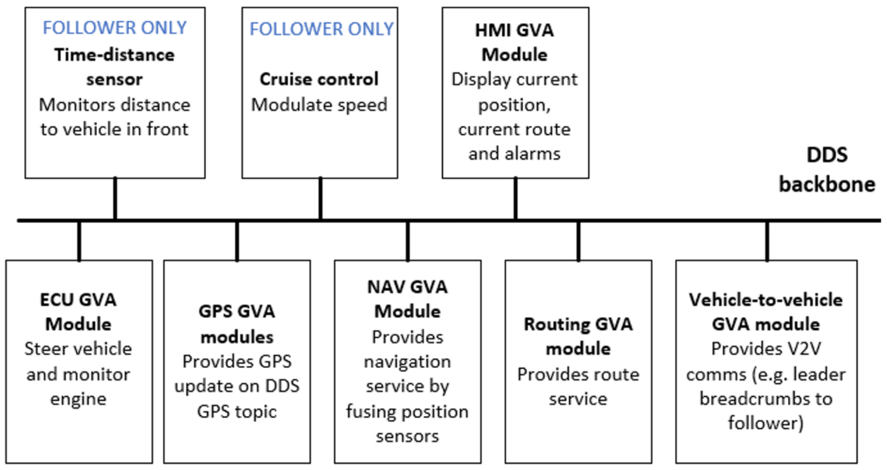
FOLLOWER ONLY
Time-distance sensor
Monitors distance to vehicle in front
FOLLOWER ONLY
Cruise control
Modulate speed
HMI GVA Module
Display current position, current route and alarms
DDS backbone
ECU GVA Module
Steer vehicle and monitor engine
GPS GVA modules
Provides GPS update on DDS GPS topic
NAV GVA Module
Provides navigation service by fusing position sensors
Routing GVA module
Provides route service
Vehicle-to-vehicle GVA module
Provides V2V comms (e.g. leader breadcrumbs to follower)

Figure 7: Overview of vehicle architecture

Co-Decyber RL agents. The first is the DDS level Access Control List (ACL) agent, and the other is the NAV reset (which manages resetting the state of the NAV module)

- GPS, which refers to the modules which perform processing on information received through the GPS sensors. There are two GPS modules (GPS0 and GPS1) which each have a single low-level agent (ACL) attached. This agent provides control over the message flow to and from the module.

An overview of the vehicle architecture as implemented in simulation is given in Figure 7.

3.3. GPS divert scenario

The attack we have modelled involves hijacking a module on the DDS network, via a malicious insider located in the rear operating base. We assume this is possible as DDS specifications do not by default include authentication of GVA modules. The hijacked module is then able to send a series of spoofed navigation messages to cause a diversion of the leader vehicle and therefore a diversion of the entire convoy (see Figure 8).

```
ATTACK GOAL: DIVERT CONVOY
     -OR: Integrity attack through DDS msg
      (-OR: Correct course using other vehicle)
     -SAND: Hijack node
          -OR: Hijack publisher (GPS topic)
          -OR: Add malicious node
     -SAND: Reset ODOM or IMS
     -SAND: Start GPS attack messages
          (-OR: Stop writing to GPS topic)
                (-SAND: Isolate using ACL)
                (-SAND: Reset NAV)
           (-OR: Control power to module)
                (-OR: Switch off MEDIA)
                (-OR: Switch off NAV)
           (-OR: Isolate switch port)
                (-OR: Isolate MEDIA)
                (-OR: Isolate NAV)
```

Figure 8: Attack-defence tree modelling convoy diversion by spoofing GPS messages through a compromised GVA module. Note the additional logic gate of SAND (Sequential AND) which means all leaf nodes must be fulfilled in sequential order before the condition holds TRUE.

Note that:

- We are targeting the DDS stream of messages between GVA modules rather than the GPS sensor itself.
- We assume that the usually optional DDS Security Specifications [21] have been implemented in the system to a degree, with loose global policies that allowed for an insider to hijack a node. These specifications are required to enable defensive actions by Co-Decyber.

This scenario represents significant additional complexity to our earlier work in defending against publishing of false fire alarms [1].

Firstly, detection latency is more of a significant factor since the longer the attack goes undetected, the more adrift the vehicles will become. Implications are discussed in Section 4.2. Secondly, there are two variations of this scenario which we needed to model in the simulator, *stateless* and *stateful*. This refers to if the NAV GVA module is configured with or without state:

- *Stateless*: The NAV module accepts new location updates regardless of how far the vehicle is from its current position estimate, allowing for immediate correction of course.
- *Stateful*: The NAV module implements a simple noise filter which will reject updates that are too far away from the current position estimate. The noise filter is a more realistic representation of the module, as real-world position fusion systems all have various filters used to improve noise rejection and positioning accuracy.

These two variants were modelled because of the change in the difficulty of defence against a cyber-attack, and provided the necessary stepwise levels of difficulty in the curriculum learning training regime for the MARL (this training technique is discussed more in Section 4.1.4).

In the stateless form, once the attacker is prevented from writing corrupt updates to NAV, it will recover on its own as it receives new correct data. However, if filtering is implemented (as per the stateful configuration) it is likely that the filters will reject the new corrupt updates. The recovery in this case requires cooperation between multiple agents, first to block the corrupt data, and then to recover the NAV module state, for example, by resetting. This can also be seen in the attack-defence tree modelling with the addition of the SAND (Sequential AND) gate as a defence to hijack of publisher in Figure 8.

From this, we can conclude that the stateful variant is the most representative of a situation in which we would want to see cooperative MARL. As such, we will be discussing results, ramifications and extensions using the stateful variant.

We had assumed in earlier work that the hijacked module that would most likely be attacked would be a low effort but connected pivot point in the vehicle system such as the MEDIA module. However, the hijacked module could be any of the GVA modules on the system, which could be attacked through various mechanisms.

Therefore, in addition to the stateful and stateless variants above, we have also created another scenario variation called varied attack source whereby the GPS divert attack could originate from MEDIA, GPS1 and GPS0. This also helps to add another level of variation to our datasets that would allow us to address the question of generalisation and reduce the risk of overfitting.

The mission impact of such an attack is also dependent on the level of location drift being broadcast by the hijacked node and detection latency. Implications of mission impact are discussed below in Section 3.4.

We use a stochastic hard-coded attacker in this work with varying parameters including:

- Time-to-detect
- Attacker location in the vehicle
- Attack start time
- Noise in the environment

This is written as a simple state machine which is both responsible for launching the attack(s) against the vehicle and updating the IDS. From the environment's point of view, the attacker is implemented as an additional agent.

We did not use intelligent ‘red’ agents as a trade-off against the complexity of training a MARL system and because balancing two systems learning against each other could create unwanted behaviours. For example, the red agent and Co-Decyber could become hyper-specialised to defeating each other rather than protecting generalised systems, or may otherwise game the system in unexpected ways. We further explore implications (including overfitting) in Section 4.1.

3.4. Mission level impacts and reward structure

From a cyber-defence point of view, attacker and defender have system level goals (attack target vehicle and mitigate attack). However, a “good” choice made by Co-Decyber can only be judged based on the mission level.

At system level, sending GPS divert messages affects our vehicle by compromising the navigation, but there is a higher level of adversarial intent, which we can assume here to be the possible intent to divert the cargo to an alternative enemy-desired destination (i.e. a mission level impact).

This mission level impact is the end-goal attacker and defender wish to achieve in a given context. This is important to consider due to three factors:

- Cascading impacts through the system: achieving only system-level defender goals could still result in mission failure (e.g., excessive security processing leading to a denial of service)
- With the multi-agent architecture, a dense reward structure (and having to consider the value of every action and associated reward for each timestep of a simulation for each agent) would be challenging, with no guarantee that the path of actions leading to the highest dense reward corresponds to the best mission objective.
- Mission level goals should be the highest priority and therefore should form the foundation of the reward structure. The more of a defender’s mission that is achieved, the better the choices that Co-Decyber is judged to have made. Note that although mission level goals have priority, that system-level goals may still influence (e.g., a small delay at the system level before defensive actions kick in).

To represent the situation above, a sparse reward structure is used. This allows the models to optimise for best mission outcome without putting artificial constraints on the process, and directly incentivises agents towards the desired mission objective. We note that sparse rewards representing mission level feedback may give too little feedback to the RL agents. We mitigate this using an offline training approach (see Section 4.1).

The reward structure for our work is shown in Table 1.

Reward	Timeliness	Cargo status
5	For every vehicle on time	With cargo
2	For every vehicle on time	Without cargo
4	For every vehicle arriving late	With cargo
2	For every vehicle arriving late	Without cargo
0	No arrival	-

Table 1
Reward structure for training

The total reward is a sum of whichever conditions hold true, for example:

- One vehicle that arrives without cargo (on time or late) is given a 2.
- One vehicle that arrives without cargo (on time or late) is given a 2.
- Two vehicles arriving delayed, with one cargo intact would be given 6.

Cargo is simulated as having a percentage chance of being stolen if the delay or diversion is significant enough. These numbers are currently arbitrary. As we move towards the real-world implementation, we envisage adding more real-world metrics (such as whether defensive actions would affect availability of the system) as well as a less binary interpretation between different outcomes (to represent scalar differences such as being a little bit late versus extremely late).

3.5. Generic framework and architecture

Vehicles contain distributed systems. As such, we chose an architecture for Co-Decyber that reflects this distribution (see Figure 9), with a Co-Decyber node coupled to and defending a unit of functionality (i.e. a GVA module). Each GVA module therefore contains a Co-Decyber node, with each node containing $1 \ldots n$ RL agents.

The framework also allows for communication not just between RL agents within a node, but also between nodes themselves. This not only allows for cyber-defence at the individual module level, but globally. As further work, we also plan to extend so that communications can also happen across systems thereby providing protection to the entire platoon. The decentralised nature of Co-Decyber makes coordination between agents more difficult, however this makes the framework more robust, since defeating Co-Decyber now involves compromising or disabling many more nodes.

We have also made design choices that would ensure a smoother transition between simulation and the real world as soon as the concept matures enough for that transfer. At a high level, this means that we:

- Ensure that partial coverage is accounted for (e.g. where only some system elements are covered by an agent or where it may not be possible to implement Co-Decyber because of system elements).
- Ensure that Co-Decyber can handle changes in available modules (i.e. modules may be added or removed from the system over time, or where modules may be compromised or isolated because of a defensive actions).
- Ensure that attack detection reflects some element of real-world performance (i.e. that it is not perfect).

4. Training

In this section we describe our training approach in Section 4.1 and expand on how our RL agents fit within the training framework in Section 4.2.

4.1. Training approach

For this work we have used Q-learning with Deep Q Networks (DQNs). DQNs have the ability to generalise and tackle large state spaces. Q-learning has also been studied in a MARL context providing further support for its use. DQNs are also well established in both research and industrial domains. All the above makes this approach a good foundation for development.

To manage system and training complexity, we have adopted several novel strategies regarding training methodology: offline RL, agent matrix training, simulation generation and curriculum learning. These are described below.

4.1.1. Offline RL

The offline RL process splits work into two decoupled phases (Figure 10) [22]. Firstly, the scenario exploration phase where different situations are set up and agents allowed to explore. The logs of these explorations are collected into an "experience database." During these simulation runs, the parameters of the agent's network are frozen, meaning that they do not learn during this phase.

Next comes the training phase, during which agents learn. In this phase, the exploration logs in the experience database are curated and used to train an improved variant of an agent.

These two phases are run in an iterative fashion to improve agents over time. By decoupling the two phases, the agent has a large and diverse database to learn from. There are several advantages [23]:

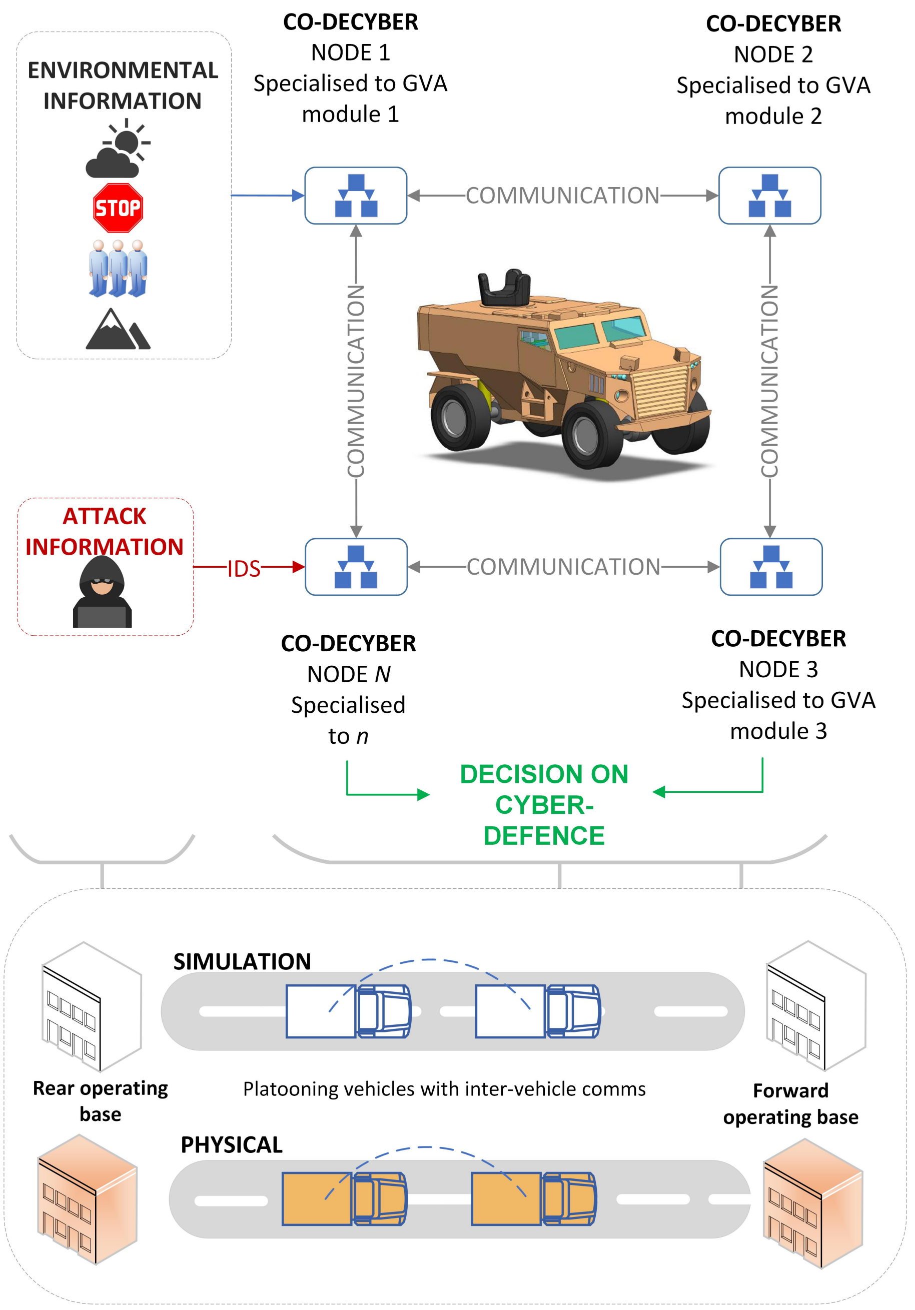
ENVIRONMENTAL INFORMATION
STOP
CO-DECYBER NODE 1 Specialised to GVA module 1
CO-DECYBER NODE 2 Specialised to GVA module 2
COMMUNICATION
COMMUNICATION
COMMUNICATION
COMMUNICATION
ATTACK INFORMATION
IDS
CO-DECYBER NODE *N* Specialised to *n*
CO-DECYBER NODE 3 Specialised to GVA module 3
DECISION ON CYBER-DEFENCE
SIMULATION
Rear operating base
Platooning vehicles with inter-vehicle comms
Forward operating base
PHYSICAL

Figure 9: The Co-Decyber framework

- Firstly, this approach reduces the interface requirements between simulation and model training systems; this is unlike an online RL approach where both systems are tightly coupled and must run simultaneously, which leads to a training system sitting idle whilst exploration progresses.
- Secondly, the experience database can, in effect, be used as a labelled dataset (and reused as needed). This also helps to mitigate the increased amount of exploration needed to deal with a sparse reward structure as mentioned in Section 3.4).
- Finally, the format, size and quality of the data within this database is flexible depending on the scenarios explored. Furthermore, optimisation of policies can happen through policy guidance from this offline experience, without having to perform exceptionally large numbers of explorations. This also mitigates against sparse reward issues.

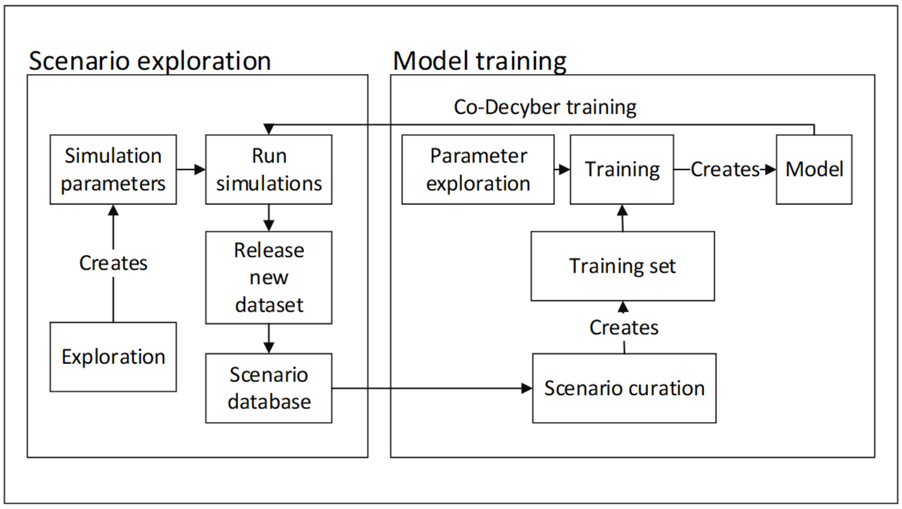

Figure 10: Overall training flow

4.1.2. Matrix training

Agents in a MARL architecture can learn to rely on each other too much, which leads to a fragile system [24]. We mitigate against this by ensuring that any given agent is trained against a variety of other agents so that they do not become overly dependent on any particular behaviour.

In practice this means that we created training datasets which include a variety of different configurations and versions of agents.

4.1.3. Simulation generation for high-quality data

The simulation environment has many different parameters that can be used to vary behaviour, broadly categorised into:

- Environmental (e.g. attacker task, vehicle locations, GPS noise etc.).
- Difficulty (e.g. NAV error tolerance, how long a fire lasts before damage occurs, etc.).
- Architectural (e.g. which agents are deployed or where an attacker is located).
- Agent (e.g. which agent variant is used as the deployed agent).

We use a generation system that samples different combinations of parameters to generate many different versions of a given scenario. We can then set an upper bound on the number of samples to draw from the available combinations. To streamline this process, we have standardized templates used to set exploration ranges for these parameters, which means that whenever new data is generated it includes a variety of different situations for the agent to learn from.

4.1.4. Curriculum learning

This is an RL training approach where the agent is presented with incrementally increased difficulty to help agents learn more complex behaviour[25].

We implement this curriculum across multiple training sessions rather than in a single run, starting with simple scenarios (e.g. beginning with a false fire alarm, moving on to a stateless GPS divert, to a stateful one, with varied attack sources). The data from those agents can then be collated into a training set to train a new set of agents that can learn more complex behaviour, building on the behaviour of the previous agents. This incremental approach allows us to assess how well agents are learning as we can compare the new agents with prior versions to see if they can tackle more situations.

4.2. Co-Decyber agents

Co-Decyber is designed to be composable so that different agents can be combined to form different types of Co-Decyber nodes.

The input data for these agents come from our representative intrusion detection system (IDS). This is implemented as a publication of detection messages as part of our attack model and is hard coded into our scenarios to reduce integration issues and the need to develop an actual detection system.

Attack detection systems in operational technology are still generally low maturity [26] and because of this, there are significant unknowns around typical IDS concerns such as false positive rates and detection latency. Because the IDS serves as input to Co-Decyber, any changes to the IDS and its performance could have significant impact on output. As such, as part of this work, we modelled different qualities of attack detection system (such as varying the time of detection) and conducted more simulation runs to cover these states to mitigate the risk. This also helps to address any potential overfitting risk.

Every low-level agent is a fully connected neural network. The agent takes input from our representative IDS, and has two types of output:

- The policy function $Q_s(a) \forall a \in A$ where the Q function describes the discounted expected future rewards from taking action a in state s. By choosing $\max_a Q_s(a)$ the expected reward is maximised. The agent does not calculate the state explicitly, but instead it is implicit in the policy function that the agent returns.
- The situation assessment output which attempts to predict the total reward the agent will receive at the end of the scenario. By adding this output, the agent is expected to learn a long-term assessment of the situation.

The agents and their corresponding defensive actions that we have currently defined are ACL, ISOLATE, FIREWALL, POWER, NAV_RESET. An outline of their actions is given below in Table 2.

Agent	Description
Access Control List (ACL)	One of the security extensions to DDS which allows the system designer to create a policy for which topics a module is allowed to access (both read and write). It is assumed that it would be possible to extend this support to allow additional block rules to be injected at run time to provide additional security. This agent can block all messages from a particular participant to the GVA module it is attached to. It can also block a particular participant from writing to a specified topic from the GVA module's point of view.
ISOLATE	Disables or enables the switch port for the GVA module that it is attached to.
FIREWALL	Blocks all messages from a specified source address or blocks messages from a specified source address to a specified destination address.
POWER	Removes power from any GVA module.
NAV_RESET	Allows Co-Decyber to reset the internal state of the NAV module. This is important because the navigation module contains the position fusion algorithm. If the attacker manages to corrupt the state of this module by injecting false data, then the vehicle may drive to the wrong location. By resetting the internal state of this module (once the false message stream has been isolated) the vehicle should recover its position and return to its proper course.

Table 2
Agent types and their purpose

5. Results

There is a substantial combinatorial challenge when assessing agent performance because there are so many models, in so many locations in the system, with many simulation variations and variants of scenarios. To address this, we have created an evaluation process which allows us to look at performance of these agents systematically.

The model evaluation flow (per scenario) begins with running the control setup, which has all defences that Co-Decyber may action disabled. Then for each low-level model, we select the best performing to run the single low-level agent setup. For each aggregation model, we select high performing low-level models and run the aggregation agent setup. For multi-node scenarios, we select high performing models (which can be low-level or aggregation models depending on whether it's a single agent Co-Decyber node or multi-agent Co-Decyber node) and run the multi-node setup. An overview can be seen in Figure 11

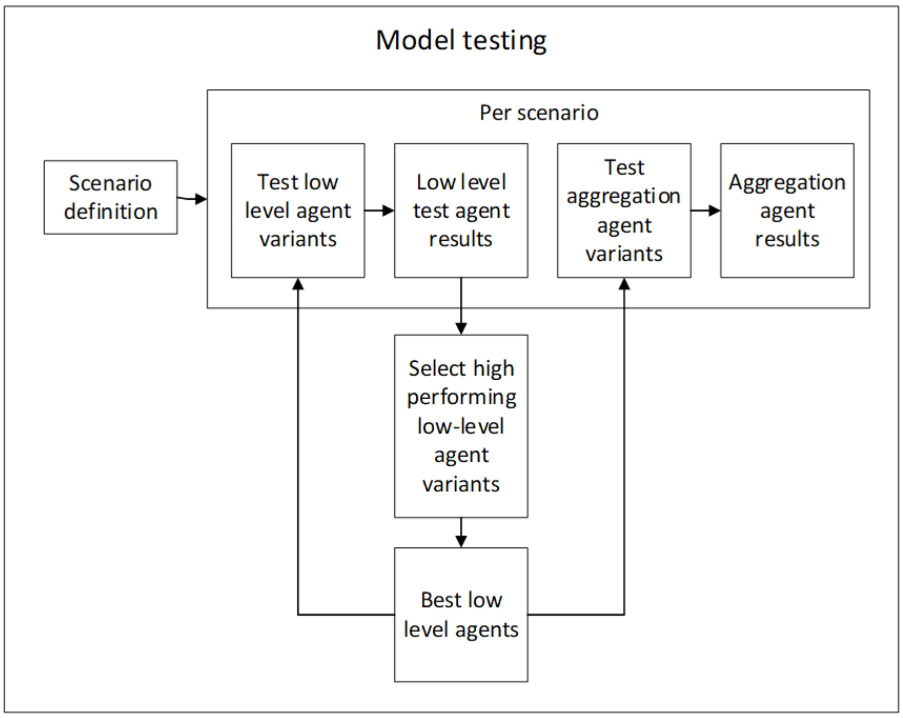
Model testing
Per scenario
Scenario definition
Test low level agent variants
Low level test agent results
Test aggregation agent variants
Aggregation agent results
Select high performing low-level agent variants
Best low level agents

Figure 11: High level model evaluation process

Each model is assessed on the average peak reward it received, which is calculated as:

- For any given run of the simulator, we choose the maximum reward achieved on any single step.
- Across multiple equivalent runs, we average them together.

This analysis approach is appropriate because of the mission level reward structure in use. It was also chosen to avoid issues where longer running simulations achieved higher rewards just because they ran for a longer period of simulated time. Initial results are presented as a simple mean across different equivalent runs since standard deviation of these results is low.

Single agent nodes, i.e., nodes that contain only one of ISOLATE, POWER CONTROL, ACL and FIREWALL all performed well within the attack scenario, gaining a maximum of a 5 reward (for a vehicle to arrive on time with cargo) 100% of the time.

We have also proven that multiple nodes can work together to defend against the attack (ISOLATE in one node and NAV_RESET in another), even when those attacks originate from varying parts of the system (labelled as "Varied Attack Scenario" or VAS – see Section 3.3). These results can be seen in Figure 12 and Table 3 for one vehicle.

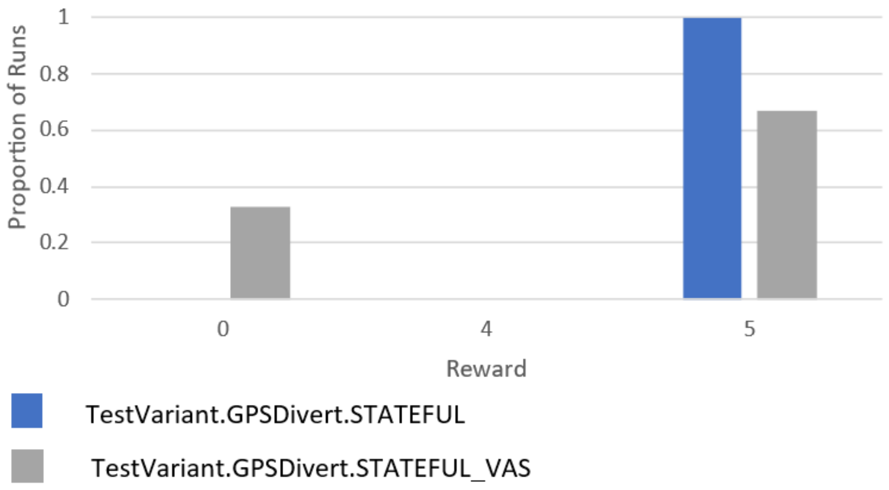

Figure 12: This shows two scenario variations (The GPS divert attack, both stateful, but one where the attack came from different locations, i.e. a varied attack source or VAS) where the maximum reward of 5 was achieved demonstrating a successful defence against an attack. This is particularly noteworthy since it involved two Co-Decyber nodes working together (one containing an ISOLATE agent, the other containing a NAV_RESET). The discrepancy (0 reward) is from an errant choice on GPS0 as can be seen in Table 3

Input	Action
ISOLATE	
Nothing	Isolate MEDIA
Attack from GPS0	Isolate distance sensor
Attack from GPS1	Isolate GPS1
NAV_RESET	
Anything	Reset NAV state

Table 3
Inputs and actions for multi-node defence

We have also visualised the defence using track data taken directly from the Co-Decyber environment, where we can clearly see that without Co-Decyber, the GPS spoof messages cause the estimate of the vehicle to be far adrift of actual vehicle position, which diverts the vehicle to an attack destination (Figure 13).

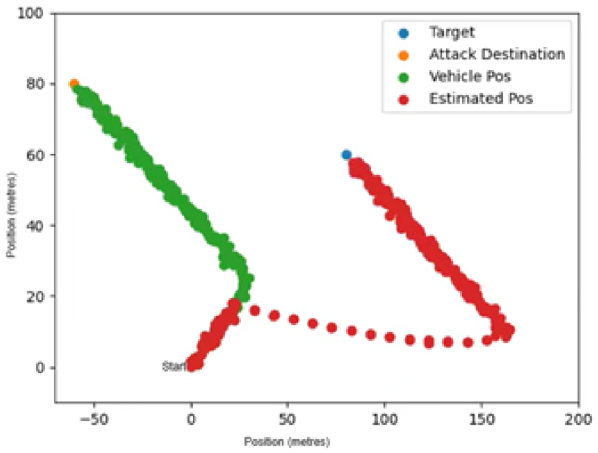

Figure 13: Attack on vehicle with no Co-Decyber, where attacker has successfully diverted the vehicle.

With Co-Decyber (this time using the FIREWALL and NAV_RESET agents), the attack occurs (which causes the red dotted blip), but the defence system successfully recovers the situation, and the vehicle continues towards its intended destination (Figure 14).

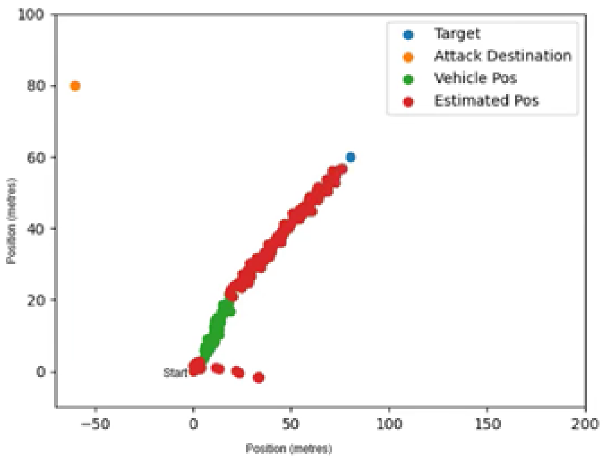

Figure 14: Attack on vehicle with Co-Decyber in place (FIREWALL and NAV_RESET), where defender has successfully recovered from the cyber attack.

6. Discussion

We have shown that the Co-Decyber framework is successful in defending against an attack. Extending from our earlier work, Co-Decyber is in fact also capable of defending against different kinds of attacks with generalised agents able to handle both a false fire alarm and the GPS divert attack. However, there are still challenges. Here we discuss two aspects: issues around generalisation (Section 6.1) and briefly outline real world considerations (6.2).

6.1. Generalisation

We recognise generalisation as a significant challenge, both in this work and in MARL research more generally. One of the first behaviours learned by all agents was what we called "pre-blocking", whereby agents had learned to disable or block the GVA module and its connections at T0 (before any actions could take place). This led to the attack never having a chance to work, leading to subsequent learning by other agents being unnecessary. This ultimately led to agents that learnt to do nothing.

From a cyber-defence point of view, this may be useful in that it successfully prevents an attack from happening, but from a MARL point of view, it meant we could not demonstrate the full power of co-operative decision making. This is a difficult problem to balance because agents are learning a valid policy, which we do not completely want to invalidate. However, it is also a static policy. We have addressed this through a combination of techniques.

Firstly, we improved our datasets, particularly in relation to data sparsity since 'interesting' events could be drowned out by the 'nothing happening' events within our simulation.

Secondly, we addressed dataset imbalance. In total there are 288 parameter combinations of which we take 50 random samples. Depending on the specific model and training set combinations, there can be significant sensitivity to hyperparameter selections. It is not unusual for only 1 of the 50 combinations to give good results. We therefore also re-evaluated our hyperparameter selections to ensure that we maximised the chances of gaining a good outcome. Some examples include:

- removing very high values of gamma (0.999) since they never gave performant results,
- changing the balance of weights for the model's two output heads (the Q function and the total reward prediction) to one where the two heads are weighted equally. We had expected this latter configuration to sacrifice some Q function performance in favour of the total reward head,

however we found this was not the case. When using the balanced loss function we got good performance for both output heads,

- higher target model update frequencies can lead to worse results (likely because of instability). We tried two values, updating every 10 epochs and updating every 20 epochs. Using the lower frequency (every 20 epochs) resulted in better Q function loss values, likely because the network had longer to learn the distribution before it was updated again leading to less spikes. Using the value of 10 also produces good results but has more variability in loss values.

We also increased the observation space. For example, we added the observations of the current isolation state of all modules (in addition to the IDS detection) to any given ISOLATE agent. This allowed the agent to learn IF-THEN policies combining detection and isolation state; for example, if the IDS detects an attack on GPS0, and GPS0 is not isolated, then isolate GPS0. We extended this to POWER and FIREWALL.

Finally, where the state space was particularly large (specifically the ACL agent where there are many more deployments in the vehicle than other agents), we simplified our approach by training ACL agents for specific nodes (Figure 15).

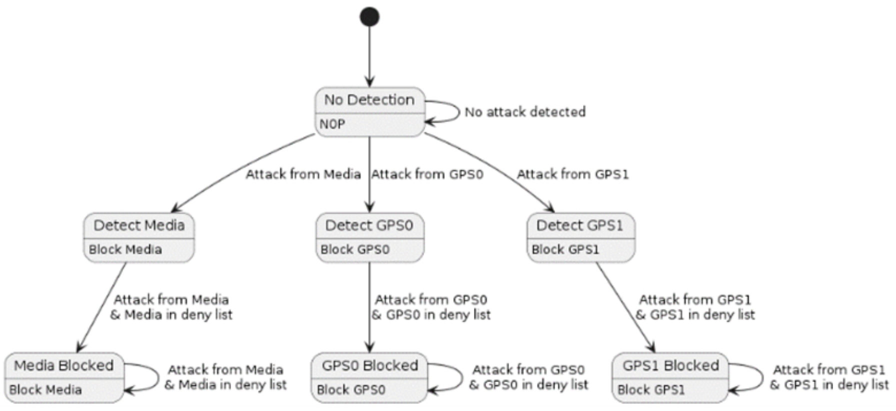
No Detection
NOP
No attack detected
Attack from Media
Attack from GPS0
Attack from GPS1
Detect Media
Block Media
Detect GPS0
Block GPS0
Detect GPS1
Block GPS1
Attack from Media & Media in deny list
Attack from GPS0 & GPS0 in deny list
Attack from GPS1 & GPS1 in deny list
Media Blocked
Block Media
GPS0 Blocked
Block GPS0
GPS1 Blocked
Block GPS1
Attack from Media & Media in deny list
Attack from GPS0 & GPS0 in deny list
Attack from GPS1 & GPS1 in deny list

Figure 15: The policy that a specialised ACL follows for the Co-Decyber node sitting on the NAV GVA module, with observations and actions taken.

6.2. Real world considerations

Finally, for the simulation-to-reality transition, Co-Decyber shows promise in terms of edge deployment. Our trained system is currently running at 10Hz (equivalent to 10 decisions per second). We aim to keep this decision rate as we continue to move from simulation into the real world, targeting a bare CPU deployment (without GPUs or accelerators) as the compute envelope.

7. Conclusion and future work

We are planning to extend this work in several ways. Firstly, we are envisaging introducing more attack scenarios as part of curriculum learning, including modelling denial of service on the vehicle-to-vehicle link, and attacking the time-distance sensor to start to introduce the agents to physical recovery actions (as opposed to cyber-actions such as isolating or firewalling) including slowing down and braking. This would allow us to continue to explore generalisation and learn how such a framework may work with the underlying vehicle controls.

Secondly, we will be updating our reward structure to include a "system availability" metric. This reward should encourage agents to keep as much of system functionality enabled as possible. We intend to implement this by extending our simulator to include tracking which parts of the system are online and able to communicate during a simulation run.

Finally, we are roadmapping possibilities in progressively making the system more realistic, including use of digital twins as a next step, through to implementing the framework in real-world representative scaled vehicles. This will give us greater knowledge and understanding of edge deployment, physical requirements, and potential real-world interactions (albeit in a demonstration environment).

In conclusion, we have shown that Co-Decyber as a framework is able to deal with our more complex GPS divert cyber-attack and is also able to generalise across both this simulated attack and the false fire alarm attack in our earlier work. This demonstrates the viability of using such a framework to assist in autonomous cyber-defence of vehicles.

Acknowledgments

This research is funded by Frazer-Nash Consultancy Ltd. on behalf of the Defence Science and Technology Laboratory (Dstl), an executive agency of the UK Ministry of Defence. The research forms part of the Autonomous Resilient Cyber Defence (ARCD) project within the Dstl Cyber Defence Enhancement programme.

References

[1] M. Cheah, J. Stone, P. Haubrick, S. Bailey, D. Rimmer, D. Till, M. Lacey, J. Kruczynska, M. Dorn, Co-decyber: Co-operative decision making for cybersecurity using deep multi-agent reinforcement learning, in: European Symposium on Research in Computer Security, Springer, 2023, pp. 628–643.

[2] N. Dhir, H. Hoeltgebaum, N. Adams, M. Briers, A. Burke, P. Jones, Prospective artificial intelligence approaches for active cyber defence, arXiv preprint arXiv:2104.09981 (2021).

[3] S. Vyas, J. Hannay, A. Bolton, P. P. Burnap, Automated cyber defence: A review, arXiv preprint arXiv:2303.04926 (2023).

[4] R. A. Bridges, A. E. Rice, S. Oesch, J. A. Nichols, C. Watson, K. Spakes, S. Norem, M. Huettel, B. Jewell, B. Weber, et al., Testing soar tools in use, Computers & Security 129 (2023) 103201.

[5] R. Jhawar, S. Mauw, I. Zakiuddin, Automating cyber defence responses using attack-defence trees and game theory, in: European Conference on Cyber Warfare and Security, Academic Conferences International Limited, 2016, p. 163.

[6] B. Kordy, S. Mauw, M. Melissen, P. Schweitzer, Attack–defense trees and two-player binary zero-sum extensive form games are equivalent, in: Decision and Game Theory for Security: First International Conference, GameSec 2010, Berlin, Germany, November 22-23, 2010. Proceedings 1, Springer, 2010, pp. 245–256.

[7] T. Eom, J. B. Hong, S. An, J. S. Park, D. S. Kim, A framework for real-time intrusion response in software defined networking using precomputed graphical security models, Security and Communication Networks 2020 (2020) 7235043.

[8] O. Gadyatskaya, Automating defence generation for risk assessment, in: 1st IEEE European Symposium on Security and Privacy, EuroS&P 2016, IEEE, 2016, pp. Poster–6.

[9] A. Tabebordbar, A. Beheshti, B. Benatallah, M. C. Barukh, Adaptive rule adaptation in unstructured and dynamic environments, in: Web Information Systems Engineering–WISE 2019: 20th International Conference, Hong Kong, China, January 19–22, 2020, Proceedings 20, Springer, 2019, pp. 326–340.

[10] T. T. Nguyen, V. J. Reddi, Deep reinforcement learning for cyber security, IEEE Transactions on Neural Networks and Learning Systems 34 (2021) 3779–3795.

[11] A. Chowdhary, D. Huang, A. Sabur, N. Vadnere, M. Kang, B. Montrose, Sdn-based moving target defense using multi-agent reinforcement learning, in: Proceedings of the first International Conference on Autonomous Intelligent Cyber defense Agents (AICA 2021), Paris, France, 2021, pp. 15–16.

[12] Q. Yao, Y. Wang, X. Xiong, P. Wang, Y. Li, Adversarial decision-making for moving target defense: a multi-agent markov game and reinforcement learning approach, Entropy 25 (2023) 605.

[13] A. Kott, Intelligent autonomous agents are key to cyber defense of the future army networks, The Cyber Defense Review 3 (2018) 57–70.

[14] A. Wilson, R. Menzies, N. Morarji, D. Foster, M. C. Mont, E. Turkbeyler, L. Gralewski, Multi-agent reinforcement learning for maritime operational technology cyber security, arXiv preprint arXiv:2401.10149 (2024).

[15] Y. Tang, J. Sun, H. Wang, J. Deng, L. Tong, W. Xu, A method of network attack-defense game and collaborative defense decision-making based on hierarchical multi-agent reinforcement learning, Computers & Security 142 (2024) 103871.

[16] I. A. Saeed, A. Selamat, M. F. Rohani, O. Krejcar, J. A. Chaudhry, A systematic state-of-the-art analysis of multi-agent intrusion detection, IEEE Access 8 (2020) 180184–180209.

[17] J. Wiebe, R. A. Mallah, L. Li, Learning cyber defence tactics from scratch with multi-agent reinforcement learning, arXiv preprint arXiv:2310.05939 (2023).

[18] B. Kordy, S. Mauw, S. Radomirović, P. Schweitzer, Foundations of attack–defense trees, in: Formal Aspects of Security and Trust: 7th International Workshop, FAST 2010, Pisa, Italy, September 16-17, 2010. Revised Selected Papers 7, Springer, 2011, pp. 80–95.

[19] G. Brockman, Openai gym, arXiv preprint arXiv:1606.01540 (2016).

[20] J. Terry, B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sullivan, L. S. Santos, C. Dieffendahl, C. Horsch, R. Perez-Vicente, et al., Pettingzoo: Gym for multi-agent reinforcement learning, Advances in Neural Information Processing Systems 34 (2021) 15032–15043.

[21] O. M. Group, About the dds security specification version 1.1, 2018. URL: http://www.omg.org/spc/DDS-SECURITY.

[22] R. F. Prudencio, M. R. Maximo, E. L. Colombini, A survey on offline reinforcement learning: Taxonomy, review, and open problems, IEEE Transactions on Neural Networks and Learning Systems (2023).

[23] H. Kiyohara, K. Kawakami, Y. Saito, Accelerating offline reinforcement learning application in real-time bidding and recommendation: Potential use of simulation, arXiv preprint arXiv:2109.08331 (2021).

[24] L. Buşoniu, R. Babuška, B. De Schutter, Multi-agent reinforcement learning: An overview, Innovations in multi-agent systems and applications-1 (2010) 183–221.

[25] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, P. Stone, Curriculum learning for reinforcement learning domains: A framework and survey, Journal of Machine Learning Research 21 (2020) 1–50.

[26] N. Jeffrey, Q. Tan, J. R. Villar, A review of anomaly detection strategies to detect threats to cyber-physical systems, Electronics 12 (2023) 3283.