LSTM-Based Anomalous Behavior Detection in Multi-Agent Reinforcement Learning

Cameron Lischke
Computer Science Department
John Hopkins university
Baltimore, MD, USA
clischk1@jhu.edu

Tongtong Liu
Computer Science Department
Wake Forest University
Winston-Salem, NC, USA
liut18@wfu.edu

Joe McCalmon
Computer Science Department
Wake Forest University
Winston-Salem, NC, USA
mccajl18@wfu.edu

Md Asifur Rahman
Computer Science Department
Wake Forest University
Winston-Salem, NC, USA
rahmm21@wfu.edu

Talal Halabi
Applied Computer Science
University of Winnipeg
Winnipeg, MB, Canada
t.halabi@uwinnipeg.ca

Sarra Alqahtani
Computer Science Department
Wake Forest University
Winston-Salem, NC, USA
alqahtas@wfu.edu

2022 IEEE International Conference on Cyber Security and Resilience (CSR) | 978-1-6654-9952-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/CSR54599.2022.9850343

***Abstract*—Multi-Agent Reinforcement Learning (MARL) extends individual reinforcement learning to enable a team of agents to collaboratively determine the global optimal policy that maximizes the sum of their local accumulated rewards. It has been recently deployed in multiple application domains such as edge computing, wireless networks, and Cyber-Physical Systems. Nonetheless, the security of MARL and its potential exposure to cyberattacks have not yet been fully investigated. This paper examines one of the most serious vulnerabilities in MARL algorithms: the *compromised agent*. This newly-engineered adversarial vulnerability is exploited when a malicious user compromises an agent to directly control its actions, and subsequently pushes its cooperative agents to act off-policy. We present a novel stacked-LSTM ensemble approach to detect such an attack. The results show that our anomalous behavior detection system significantly outperforms five baselines from the literature.**

***Index Terms*—Multi-Agent Reinforcement Learning, adversarial attacks, LSTM, anomaly detection.**

I. Introduction

The last decade has witnessed significant advances in the usage and implementation of Reinforcement Learning (RL) in a variety of application areas. Furthermore, agents learning to play card games like Go and Poker, advancements in autonomous driving systems, and a multitude of robotics innovations all center around RL algorithms that optimize the training of more than one agent in parallel. Naturally, these algorithms emerged as a subfield of RL known as Multi-Agent Reinforcement Learning (MARL).

With MARL's major implications in artificial intelligence applications ranging from computer-driven games to autonomous vehicle platoons [1], the reliability and security of MARL algorithms is of utmost importance. Because of the common goal of cooperation or competition (or both) between agents within many MARL systems [2], agents react based on the actions of other agents within the same environment. For example, connected autonomous vehicles must interact with other vehicles, pedestrians, and the static infrastructure while considering the actions and reactions of other vehicles before acting on their own [3]. For this reason, it only takes one corrupted agent to cause major security issues. Adversaries that intentionally aim to defect targeted networks can confuse agents and lead them to make mistakes that can result in poor performance and even harm to humans that rely on these systems [4], [5]. Recently, this vulnerability of compromised agents has been discovered in MARL policies in [5].

Often, the noise added by corrupted agents to fool MARL systems are invisible to humans [1], [4], which underlines the critical need for reliable machine learning algorithms to detect and mitigate such attacks. In this paper, we develop a novel ensemble of binary long-short term memory (LSTM) network and predictive LSTM to detect the compromised agent behavior in MARL. We evaluate the performance of our approach against popular classification approaches, namely four dense layers neural networks, support vector machines, k-nearest neighbors, and binary LSTM [6]. Our ensemble model achieves the highest detection rate among all baselines. We uploaded our code and datasets as supplementary materials to ensure the reproducibility of our research findings[1].

Our contributions are listed as follows :

1) We develop a novel ensemble model using binary LSTM and predictive LSTM to detect the behavior of compromised agents in MARL systems.
2) We design anomalous behavior detection models using different machine learning techniques as baselines.
3) We thoroughly analyze the detection results of the proposed model and the baselines using the attack from [5]. Our ensemble model detects up to 100% with high attack rates and outperforms the studied baselines.

The remainder of the paper is structured as follows. Section II introduces some preliminary concepts and discusses back-

[1]https://github.com/frank47ltt/Multi-Agent-Security/tree/main/MARL_Detection

ground information and related work. Section III presents our proposed detection approach. Section IV analyzes the obtained results. Finally, Section V concludes the paper.

II. Preliminaries and Related Work

This section provides important information about the security of MARL and existing anomaly detection models.

A. Multi-Agent Reinforcement Learning

MARL extends single-agent RL and employs $N > 1$ agents in a single environment. MARL environments are not inherently Markov, since the transition probabilities $Pr\{R_{t+1} = r, S_{t+1} = s' | S_t, A_t^1, \dots, A_t^N\}$ depend not only on one action but on the actions chosen by each agent. To each agent, the function by which the environment selects the state at $t+1$ appears non-stationary [7]. In addition, each agent receives its own observations, o_t^i, where $S_t = \{o_t^1, o_t^2, \dots, o_t^N\}$. Thus, parts of the environment's state can be hidden from each agent.

MADDPG is the current state-of-the-art algorithm for MARL. It uses two deep neural networks, one to learn a value representation for the expected return (the critic), and one to decide the correct action for the agent (the actor). MADDPG allows the critic network to learn a value function based on the observations of each agent combined, but the actor network must learn a policy using only the corresponding agent's own observations. The result is an agent which has a unique decision-learning policy from the others in the environment, allowing success for multiple agents with different goals.

The other MARL algorithm that we tested in this paper is QMIX [8], which is a multi-agent Deep Q-Learning (DQN) algorithm based on Q-Learning with state-of-the-art performance on the StarCraft II Multi-Agent Challenge. QMIX finds the optimal joint action value function using a monotonic mixing function of per-agent utilities. In QMIX, each agent uses a recurrent-DQN network to estimate the action values based on their partial observation. DQN uses experience replay and a target network to improve the stability and convergence of the RL agent's training. To maximize the total team reward, QMIX estimates the joint action values through a mixing network that takes in each agent's selected action Q-value (i.e., action with max Q-value) and the current state to estimate the total team reward. With an accurate estimation of the total reward, each individual agent's Q-network can be fine-tuned to maximize the total team reward during execution.

B. Adversarial Attacks in MARL

Research surrounding MARL systems has been gaining a lot of popularity [9]. The heightened interest in RL systems has brought significant attention to the adversarial vulnerabilities evident in many cooperative and competitive settings [3]. Huang et al. [4] were the first to present the proof of concept that an adversary can interfere with the operation of an RL agent [4]. In an attempt to coordinate its actions with other agents in the system, an uncompromised agent can be directly influenced by its malicious adversarial agents [2].

To detect and mitigate such attacks that are normally invisible to the human eye [1], [4], machine learning techniques are especially valuable. Much of the research involves investigations of how to prevent adversarial examples from fooling MARL policies with methods like adversarial training [1], [10], [11], data augmentation and randomization [12], and detector subnetworks [1], [13]. In this work, we test several popular machine learning approaches and their ability to detect adversarial attacks when they have already occurred.

C. Deep Learning for Anomaly Detection

Deep learning and the use of neural networks to conceptualize tasks have become hot topics. However, due to limitations of a feed-forward neural network's ability to remember information from previous frames of data, a Long-Short Term Memory Neural Network (LSTM) was developed using real-time recurrent learning [14]. LSTMs mitigate the vanishing-gradient problem by implementing various recurrent cycles from subsequent neurons to preceding ones, creating hidden layers that act like memory. Using input, activation, forget, and output gates, LSTM allows for long-term memory storage and is especially conducive to time-series data [6], [14].

LSTMs demonstrate a viable technique to predict normal time-series behavior that consequently classifies anomalous behavior without real knowledge of the data domain [14], [15]. Compared to other deep learning techniques, LSTM-based prediction models may yield better results and performance [6]. Much work has been done to prove that LSTMs are extremely effective in classifying anomalous behavior [13] in various domains ranging from medicine [14] to computer network traffic [15]. LSTM networks can go even further, differentiating between categories of anomalous behavior from normal behavior [14]. Using prediction error distributions as a baseline [6], LSTM networks are perfectly suited to analyze sequential data with temporal dynamics [14]. Stacked LSTM neural networks have shown exceptional performance in detecting anomalies [6], [14], [16], demonstrating the promise of deep learning techniques in security applications.

Malhotra et al. [6] claim that LSTM-based prediction models, where the output is the expected action or value, may produce better results than other detection subnetworks. Their detection algorithm learns a threshold that maximizes the F1-score between anomalous and normal datapoints. Then, after training on normal datasets, an LSTM prediction model calculates the error vector of predicted and true values. Fitting this to a multi-variate Gaussian distribution gives an output of P^t; if $P^t < threshold$, the data will be classified as anomalous. Naseer et al. [16] propose a simpler deep learning detector subnetwork using a binary LSTM model, which classifies datapoints into 0 as normal and 1 as anomalous. LSTM-based models have shown to outperform other machine learning classifiers. Overall, the temporal element of MARL systems can clearly benefit from LSTM-based detector subnetworks.

D. Classical Machine Learning for Anomaly Detection

In this paper, we develop anomaly detection models to detect the compromised agent in MARL using SVM, K-nearest, Random Forests, binary LSTM, and four layer dense neural networks. Then, we compare their performance against our novel detection model of ensemble binary LSTM and predictive LSTM. The baseline models are described as follows.

Support Vector Machines (SVMs) [17] are one of the most common machine classifiers for binary data. The inner workings of these machines relies on input vectors, a distance metric hyperparameter, and labeled points. Assuming that the data can be linearly separable, a linear decision surface is constructed and used to classify data by finding the best (largest) hyperplane that separates all data points of one class from those of another class, without interior data points. Once that decision boundary is learned, datapoints are plotted and then classified based on which half of the decision boundary it falls into. Basic SVMs can reliably classify anomalous data due to its binary nature, assuming that the training data are independently and identically distributed [18]–[20].

k-nearest neighbors is another supervised learning method. Using this algorithm, predictions are made for a new instance by searching through the entire training set for the k most-similar instances (neighbors) and summarizing the most-common output label for those k neighbors. In short, the class with the highest frequency from the k-nearest neighbors will be taken as the new datapoint's predicted class. Again, a distance metric hyperparameter, as well as an integer for k must be specified prior to testing. As in SVM, a popular application for k-nearest neighbors is network intrusion detection. Because of the distance metric used to determine the nearest neighbors, high-dimensional vectors that are similar may not actually be regarded as "close." In addition, as the number of training datapoints increases exponentially, the probability of error is bounded [21]. To combat this, k-nearest neighbors is often combined with more sophisticated algorithms [22].

In other application settings, random forests have proven useful. The basic unit of a random forest is the decision tree, an overall map of the possible outcomes of a series of related choices, usually beginning with a single node and repeatedly branching into possible outcomes until a classification can be made by the tree. Randomly creating a forest of trees presents an ensemble that forms a random forest providing greater accuracy than stand-alone decision trees by averaging the predictions of each component tree [14], [23]–[26]. Generally, with more trees and stronger correlation between them comes greater robustness and better classification [14], [23].

III. Proposed Anomaly Detection Approach

This section describes the threat model in MARL based on the compromised agent vulnerability and presents our anomalous behavior detection model.

A. Threat Model

This paper models the cooperative MARL (MARL) system using stochastic games [27]. For an n-agent stochastic game, we define a tuple $G = \big(S, A^1, ..., A^n, r^1, ..., r^n, T, \gamma\big)$, where S denotes the state space, A^i and r^i are the action space and the reward function for agent $i \in 1, ..., n$, respectively. γ is the discount factor for future rewards, and T is a joint state transition function $T : S \times A_1 \times A_2 .. \times A_n \to \triangle(S)$ where $\triangle(S)$ is a probability distribution on S. Agent i chooses its action $a^i \in A^i$ according to its policy $\pi^i_{\theta^i}(a^i|s)$ parameterized by θ^i conditioning on some given state $s \in S$. The collection of all agents' policies π_θ is called the joint policy where θ represents the joint parameter. For convenience, we interpret the joint policy from the perspective of agent i as:

$$\pi_\theta = (\pi^i_{\theta^i}(a^i|s)\pi^{-i}_{\theta^{-i}}(a^{-i}|s)) \tag{1}$$

where $a^{-i} = (a^j)_{j \neq i}, \theta^{-i} = (\theta^j)_{j \neq i}$, and $\pi^{-i}_{\theta^{-i}}(a^{-i}|s)$ is a compact representation of the joint policy of all complementary agents of i. Since the agents' policies are held fixed, the Markov game G reduces to a single-player MDP, denoted by $G_m = (S, A_m, T_m, R_m)$, that the adversary must solve to generate a new policy by which the compromised agent m will achieve the adversary's goal of attacking the other agents $-m$ and degrading the robustness of MARL.

The compromised agent problem in MARL is defined by:

$$\max \sum_{t=0}^{t=T} \mathbf{KL} \;\, (p(a_t^{-m}|a_t^m, s_t) || p(a_t^{-m}|a_t^{*m}, s_t)) \tag{2}$$

where a^{*m} represents the adversarial actions generated by the adversarial policies of the compromised agent m. This equation maximizes the KL-divergence between the conditional policy of $-m$ on the action a^m at time t and the same conditional policy if agent m deviates from its policy and takes an adversarial action a^{*m}. The adversary can then intervene on a_t^m by replacing it with action a_t^{*m}, which will be used to compute the next action of agents $-m$, $p(a_{t+1}^{-m}|a_t^{*m}, s_t^m)$, pushing the activations of their policy networks off distribution. Practically, the problem in Eq. (2) can be solved by finding the compromised agent's adversarial actions a^{*m} that maximize the KL-divergence using the attack proposed in [5].

B. Anomalous Behavior Detection in MARL using an Ensemble LSTM Model

To combat the compromised agent vulnerability in MARL algorithms, we develop an ensemble model of binary LSTM and predictive LSTM shown in Fig. 1. The proposed model predicts the sequence of actions expected to be taken by the compromised agent m at the next h time steps $t + h$ based on the actions taken by the benign agents $-m$, the actions taken by m at the last $t - h$ time steps, and the states $s_t, s_{t-1}, ..s_{t-h}$. Let's assume that the prediction function for the ensemble LSTM is $\overrightarrow{y} = \omega(x)$ such that y is the prediction output and x represents the input to the network. $\overrightarrow{y}$ is a vector of the predicted next h actions for agents m, $[a_t^m, .., a_{t+h}^m]$. The input x includes three vectors of historical information: a vector for the previous h actions taken by agents m, $[a_{t-h}^m, ..., a_{t-1}^m]$, a vector for the previous h actions taken by agents $-m$ $[a_{t-h}^{-m}, ..., a_{t-1}^{-m}]$, and a vector of previous states $[s_{t-h}, ..., s_{t-1}]$. Then, the LSTM prediction function becomes:

$$\overrightarrow{a}^m_{t,t+h} = \omega^m \left([a_{t-h}^m, .., a_t^m], [a_{t-h}^{-m}, .., a_t^{-m}], [s_{t-h}, .., s_t]\right) \tag{3}$$

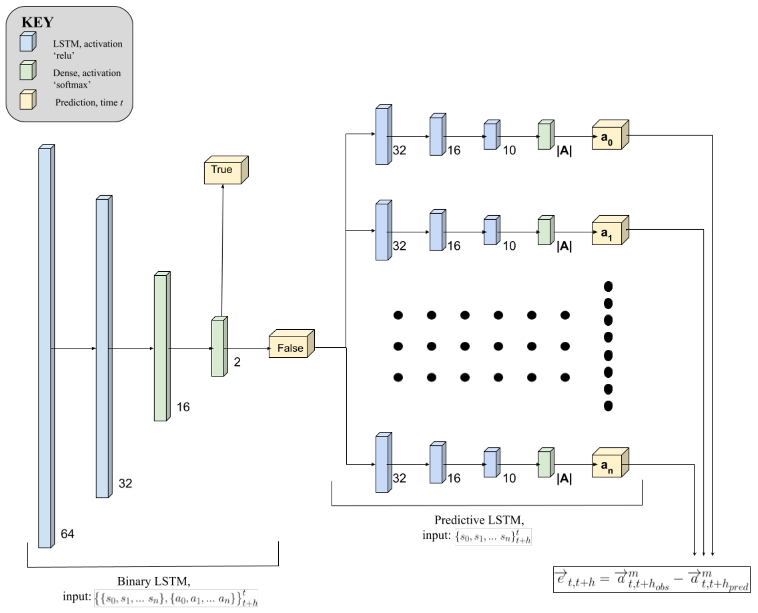
KEY
LSTM, activation 'relu'
Dense, activation 'softmax'
Prediction, time t
64
32
16
2
True
False
32
16
10
|A|
a_0
a_1
a_n
Binary LSTM,
input: $\{\{s_0, s_1, ... s_n\}, \{a_0, a_1, ... a_n\}\}_{t+h}^{t}$
Predictive LSTM,
input: $\{s_0, s_1, ... s_n\}_{t+h}^{t}$
$\overrightarrow{e}_{t,t+h} = \overrightarrow{a}^{m}_{t,t+h_{obs}} - \overrightarrow{a}^{m}_{t,t+h_{pred}}$

Fig. 1. The ensemble LSTM-based model for anomalous behavior detection in MARL.

Using the results from Eq. (3), we compute the error vector $\overrightarrow{e}$ based on Mahalanobis' distance [28] between the predicted and observed vectors of actions as follows:

$$\overrightarrow{e}_{t,t+h} = \overrightarrow{a}^{m}_{t,t+h_{obs}} - \overrightarrow{a}^{m}_{t,t+h_{pred}} \tag{4}$$

Using a predefined threshold σ, we compute how likely the error vector $\overrightarrow{e}$ represents an anomaly if it is larger than σ.

Algorithm 1 presents the LSTM model for MARL anomaly detection. In Fig. 1, we first run our testing data through the Binary LSTM; if a datapoint is classified as positive for anomaly, then it would be investigated by a human to ensure accuracy. Any datapoint classified as negative would then be run through the predictive LSTM ensemble separated by agents. These points are tested by n models to generate the predictions based on the probability that agent i performs action $1, 2, \dots, k$ for k possible actions.

Algorithm 1 Ensemble Model for Anomalous Behavior Detection in MARL

```
 1: Input
 2: $[s_0, s_1, .., s_n]_{t-h}^{t}$: Agents' states for the last h timesteps
 3: $[a_0, a_1, .., a_n]_{t-h}^{t}$: Agents' actions for the last h timesteps
 4: $D$: The training dataset from MARL systems
 5: Output
 6: $\overrightarrow{e}_{t,t+h}$: The classification of each agent's action as normal
    or anomaly
 7: procedure ENSEMBLE LSTM MODEL
 8:     Learn $\alpha$ by maximizing F1-score in D
 9:     $\overrightarrow{a}^{m}_{t,t+h} = \omega^{m}\left([a^{m}_{t-h}, ., a^{m}_{t}], [a^{-m}_{t-h}, ., a^{-m}_{t}], [s_{t-h}, ., s_{t}]\right)$
10:     $\overrightarrow{e}_{t,t+h} = \overrightarrow{a}^{m}_{t,t+h_{obs}} - \overrightarrow{a}^{m}_{t,t+h_{pred}}$
11:     if $\overrightarrow{e}_{t,t+h} > \alpha$ then
12:         return $a^{m}_{t}$ is anomalous
13:     else
14:         return $a^{m}_{t}$ is normal
```

In the tested particle environments, we used a threshold of 0.10 for the minimum probability for action j such that i performing j is still a relatively benign action (as suggested by the model). The threshold 0.10 was chosen because there are only 5 possible actions in our test environments. For the model to decide that a certain action is normal, it only needs to predict it with at least 0.20 confidence. Therefore, 0.10 has merit as a threshold. Similarly, Starcraft II has 9 possible actions, so for that environment, a threshold of 0.05 has merit. For a datapoint to be considered normal, all of the true actions must be covered by all of the predicted actions with a confidence greater than the threshold. Formally, this is defined as follows: $\forall\ \hat{a} \in true\{a_0, a_1, \dots, a_n\},\ \exists\ a \in \{predicted\{a_0, a_1, \dots, a_n\} \geq \alpha\}$ such that $\hat{a} == a$.

IV. Experiments and Results

We developed different anomalous behavior detection models as the baseline for our experimental evaluation, namely a four-layer dense neural network, a three-layer binary LSTM [16], Random Forest, SVM classifier, and a K-nearest Neighbors classifier. This section explains the dataset used in our empirical study and analyzes the obtained results.

A. Dataset Generation

To create the dataset for the MARL system, we first ran the two MARL algorithms MADDPG and QMIX without anomalies. In MADDPG, we collected data from 2 particle environments: cooperative navigation and physical deception. In cooperative navigation, N cooperative agents must cover

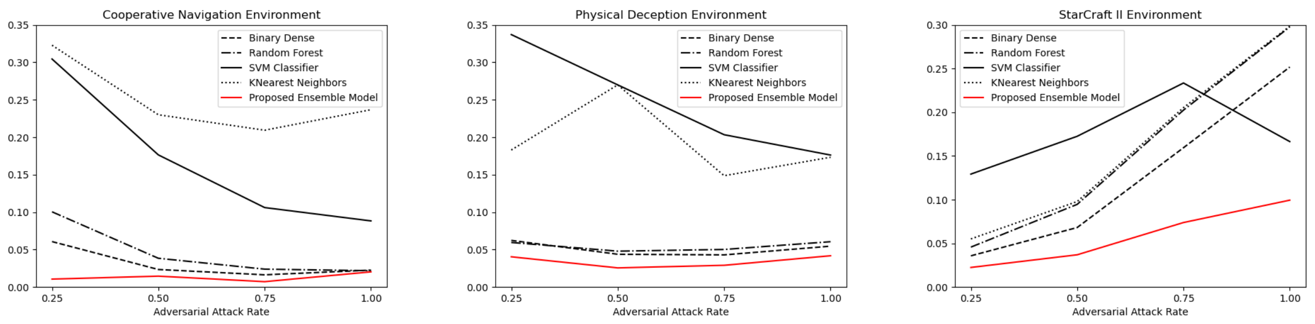

Fig. 2. The percentage of undetected anomalies by each model in the white-box compromised agent attack scenario.

L landmarks and the agents must learn to reach separate landmarks without communicating their observations to each other. In our experiments, we use $N = L = 3$. In physical deception, N cooperative agents try to fool one adversarial agent. There are L total landmarks with one being the 'target' landmark and only the cooperative agents know which of the landmarks is the target. The adversary must try to infer and reach the target landmark from the cooperative agents' positioning, and the cooperative agents must try to deceive the adversary by spacing out. The cooperative agents are rewarded as long as a single member of their team reaches the target landmark. Here, we use $N = L = 2$.

We also collected data from the QMIX algorithm using the StarCraft II environment, which is a real-time strategy game where two teams of agents can fight against each other. We use the "3m" SMAC map, which employs three "Marine" units on each team. In this scenario, a team wins by shooting the enemy team enough to where they run out of health. Our team consists of three RL agents working together in a MARL system to defeat the three enemy marines, which use fixed policies. The implemented attack in [5] controls one of the marines and alters its decision-making based on the adversairal policy.

After collecting the normal dataset, we implemented the compromised agent attack [5], during which we control the compromised agent using an adversarial policy to change its optimal actions in white and black-box settings. All of the baseline classifiers were trained using an evenly-distributed training set of approximately 50% anomalous and 50% normal datapoints. The baseline classifiers predicted abnormality based on just one single input vector of the current timestep's global observations and global actions. That is, they received the current state $\{s_0, s_1, ...s_n\}$ and actions $\{a_0, a_1, ...a_n\}$ of all agents. The binary LSTM received the same information for the three previous timesteps, as well as $\{\{s_0, s_1, ...s_n\}, \{a_0, a_1, ...a_n\}\}_{t-3}^{t}$.

B. Results and Discussion

Fig. 2 compares the proposed ensemble model against the baseline detection models using the compromised agent attack. The ensemble model achieved successful detection results in all environments based on its recall and precision metrics. Recall is the number of correctly identified anomalies in

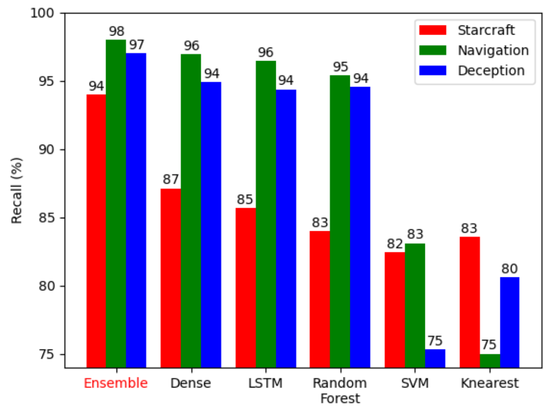

Fig. 3. Average recall across all attack rates for each of the baseline models.

proportion to the number of true anomalies in the datasets. Precision is the proportion of correctly identified anomalies in relation to the number of predicted anomalies. Hence, lower precision corresponds to higher false positive rates. The sensitive nature of the attack makes recall paramount in measuring performance. The ensemble model consistently showed the highest recall (percentage of anomalies detected), with some attacks only going undetected 0.1% of the time.

Fig. 3 emphasizes the performance of the ensemble model, particularly in the Starcraft II environment, where the average recall across all attack rates improved by approximately 7%. This large improvement in recall compared to the other baseline models in the Starcraft II environment is likely due to the high-complexity of the problem. Because of the environment's high-dimensionality, basic machine learning algorithms struggle to distinguish between normal and anomalous datapoints.

Since the proposed ensemble model yields the best performance, we highlight its detection details in TABLE I, which shows that our model consistently detects close to 100% of the attacks. The model achieves higher precision when the attack rate increases suggesting that at least 90% of the attacks can easily be traced and detected at an attack rate as low as 25% of the episode time. Nonetheless, the model precision leaves some room for improvement; we believe that this is expected

TABLE I
THE DETECTION RESULTS OF THE ATTACKS USING THE ENSEMBLE LSTM MODEL IN THE THREE ENVIRONMENTS ACROSS DIFFERENT ATTACK RATES IN BOTH WHITE AND BLACK BOX SETTINGS.

Cooperative Navigation				
	White Box		**Black Box**	
Attack Rate	Precision	Recall	Precision	Recall
25%	37%	83%	37%	91%
50%	88%	92.5%	64%	95%
75%	88%	94%	81%	95%
100%	91%	96%	100%	97%
Physical Deception				
	White Box		**Black Box**	
Attack Rate	Precision	Recall	Precision	Recall
25%	28%	85%	32%	95%
50%	92%	92%	60%	97%
75%	87%	93%	81%	98%
100%	84%	92%	100%	96%
Starcraft II				
	White Box		**Black Box**	
Attack Rate	Precision	Recall	Precision	Recall
25%	39%	91%	71%	90%
50%	96%	99%	61%	99%
75%	92%	99%	82%	85%
100%	90%	98.5%	77%	90%

from a model trained on evenly distributed data but tested on undistributed data. With higher attacking rates, the model gradually increases its precision. Like all intrusion detection systems, the emphasis should be on reducing the number of undetected attacks. Based on this reasoning, the outstanding performance of the ensemble in terms of recall is crucial.

V. Conclusion

This paper proposes an anomalous behavior detection approach for compromised agents in MARL systems. Specifically, we develop an ensemble of binary LSTM and predictive LSTM to detect the compromised agent in MADDPG algorithm using 2 particle environments and QMIX algorithm using StarCraft II in both white-box and black-box settings. We tested our ensemble model using a dataset created by implementing the compromised agent attack. Our model consistently showed the highest recall compared to state-of-the-art baseline models, with some attacks only going undetected 0.1% of the time. In the future, we will develop more complex and stealthy attacks against MARL that will allow us to further improve the robustness of our anomaly detection model.

Acknowledgment

This material is based upon work supported by the National Science Foundation (NSF) under grant no. 2105007.

References

[1] T. Chen, J. Liu, Y. Xiang, W. Niu, E. Tong, and Z. Han, "Adversarial attack and defense in reinforcement learning-from ai security view," *Cybersecurity*, vol. 2, 12 2019.

[2] S. L. Barton, N. R. Waytowich, and D. E. Asher, "Coordination-driven learning in multi-agent problem spaces," *CoRR*, vol. abs/1809.04918, 2018.

[3] S. W. Loke, "Cooperative automated vehicles: A review of opportunities and challenges in socially intelligent vehicles beyond networking," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 4, pp. 509–518, 2019.

[4] S. H. Huang, N. Papernot, I. J. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *CoRR*, vol. abs/1702.02284, 2017.

[5] J. Lin, K. Dzeparoska, S. Q. Zhang, A. Leon-Garcia, and N. Papernot, "On the robustness of cooperative multi-agent reinforcement learning," in *2020 IEEE Security and Privacy Workshops (SPW)*, pp. 62–68, IEEE, 2020.

[6] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2015.

[7] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[8] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *International Conference on Machine Learning*, pp. 4295–4304, PMLR, 2018.

[9] G. Arslan and S. Yüksel, "Decentralized q-learning for stochastic teams and games," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1545–1558, 2016.

[10] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, "Adversarial policies: Attacking deep reinforcement learning," *arXiv preprint arXiv:1905.10615*, 2019.

[11] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[12] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," *arXiv preprint arXiv:1711.01991*, 2017.

[13] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," *arXiv preprint arXiv:1702.04267*, 2017.

[14] A. Verner, *LSTM networks for detection and classification of anomalies in raw sensor data*. PhD thesis, Nova Southeastern University, 2019.

[15] R. C. Staudemeyer and C. W. Omlin, "Evaluating performance of long short-term memory recurrent neural networks on intrusion detection data," in *Proceedings of the South African institute for computer scientists and information technologists conference*, pp. 218–224, 2013.

[16] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, and K. Han, "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48231–48246, 2018.

[17] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[18] W. Hu, Y. Liao, and V. R. Vemuri, "Robust anomaly detection using support vector machines," in *Proceedings of the international conference on machine learning*, pp. 282–289, Citeseer, 2003.

[19] A. George, "Anomaly detection based on machine learning dimensionality reduction using pca and classification using svm," *International Journal of Computer Applications*, vol. 47, pp. 5–8, 06 2012.

[20] O. Salem, A. Guerassimov, A. Mehaoua, A. Marcus, and B. Furht, "Anomaly detection in medical wireless sensor networks using svm and linear regression models," *International Journal of E-Health and Medical Communications (IJEHMC)*, vol. 5, no. 1, pp. 20–45, 2014.

[21] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[22] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," *Computers & security*, vol. 21, no. 5, pp. 439–448, 2002.

[23] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. Hashem, "Attack and anomaly detection in iot sensors in iot sites using machine learning approaches," *Internet of Things*, vol. 7, p. 100059, 2019.

[24] S. Xuan, G. Liu, Z. Li, L. Zheng, S. Wang, and C. Jiang, "Random forest for credit card fraud detection," in *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*, pp. 1–6, 2018.

[25] R. Primartha and B. A. Tama, "Anomaly detection using random forest: A performance revisited," in *2017 International Conference on Data and Software Engineering (ICoDSE)*, pp. 1–6, 2017.

[26] J. Zhang and M. Zulkernine, "A hybrid network intrusion detection technique using random forests," in *First International Conference on Availability, Reliability and Security (ARES'06)*, pp. 8 pp.–269, 2006.

[27] L. S. Shapley, "Stochastic games," *Proceedings of the National Academy of Sciences*, vol. 39, no. 10, pp. 1095–1100, 1953.

[28] G. Mclachlan, "Mahalanobis distance," *Resonance*, vol. 4, pp. 20–26, 06 1999.