Combined Danger Signal and Anomaly-Based Threat Detection in Cyber-Physical Systems

Viktoriya Degeler, Richard French, and Kevin Jones(✉)

Airbus Group Innovations, Newport, UK
{viktoriya.degeler,richard.french,kevin.jones}@airbus.com

Abstract. Increasing number of physical systems being connected to the internet raises security concerns about the possibility of cyber-attacks that can cause severe physical damage. Signature-based malware protection can detect known hazards, but cannot protect against new attacks with unknown attack signatures. Anomaly detection mechanisms are often used in combination with signature-based anti-viruses, however, they too have a weakness of triggering on any new previously unseen activity, even if the activity is legitimate. In this paper, we present a solution to the problem of protecting an industrial process from cyber attacks, having robotic manufacture facilities with automated guided vehicles (AGVs) as our use case. Our solution combines detection of danger signals with anomaly detection in order to minimize mis-labelling of legitimate new behaviour as dangerous.

Keywords: Intrusion detection · Anomaly detection · Danger Theory · Automated Guided Vehicles · Cyber-Physical Systems

1 Introduction

While increasing numbers of physical systems being connected to the internet brings enormous possibilities for technological progress, it also raises huge security concerns. Cyber-Physical Systems have already been shown to be susceptible to cyber attacks that can cause (sometimes catastrophic) physical damage. The German Federal Office for Information Security (BSI) revealed in 2014 that a steel manufacturing facility suffered massive damage after it was not able to shutdown a blast furnace in a controlled manner due to malicious code implanted into its control system [3]. Earlier, the Stuxnet virus gained fame after successfully attacking programmable logic controllers of Iran's nuclear centrifuges, changing their rotation speed, which resulted in physical damage to many of them [5].

Signature-based malware protection can detect known hazards, but cannot protect against new attacks with unknown attack signatures, which is especially important due to advances of automated malware creation [1]. Anomaly detecting intelligent mechanisms are often used in combination with signature-based anti-viruses, in order to detect and prevent anomalous activity. As an example, the *negative selection* [2] approach compares all new events with a previously

B. Mandler et al. (Eds.) IoT 360° 2015, Part I, LNICST 169, pp. 27–39, 2016.
DOI: 10.1007/978-3-319-47063-4_3

constructed set of *non-self* entities, i.e. those that fail similarity tests with known *self* entities. Unfortunately, anomaly detection mechanisms have a known weakness of triggering on any new previously unseen activity. In some cases, such as fraud detection in banking systems, anomaly detection leads to great results, because it can be reasonably expected that "self" detector cover all types of legitimate behaviour and any anomaly ("non-self") is therefore a fraud. However, in many other types of systems, including internet of things, network-based ones, the behaviour changes over time and has legitimate anomalous events. Using anomaly-based threat detection in such systems will create huge amount of false positives, i.e. mis-detecting legitimate behaviour as an attack, thus disrupting normal course of operations, which can sometimes lead to economic and operational losses comparable to genuine attacks. Therefore it is important that the adaptive detection mechanism keeps both types of mistakes (false negative and false positive) at minimum. On the other hand, in Danger Theory [7], coming from Artificial Immune Systems research area [4], responses are triggered by *danger signals* rather than presence of *non-self* objects. Entities are allowed to exist until harmful signals are received. If a harmful activity (e.g. cell death) is detected, the immune response is triggered, attacking either all foreign entities, or all entities locally, depending on the severity of the danger.

In this paper, we present a solution to the problem of protecting an industrial process from cyber attacks, having robotic manufacture facilities with automated guided vehicles (AGVs) as our use case. Our solution combines detection of danger signals with anomaly detection to minimize mis-labelling of legitimate new behaviour as dangerous. In Sect. 2, we present our use case and system architecture. Section 3 provides detailed explanation of our danger detection module. Section 4 evaluates the solution and Sect. 5 concludes the paper.

2 Automated Guided Vehicles Protection

Factories with a complex manufacturing cycle often rely on Automated Guided Vehicles (AGVs) for moving materials across work cells. AGV control systems and equipment are usually networked and distributed to allow submitting tasks and operate AGVs remotely. This puts AGV control systems at risk of being exposed to cyber attacks, as has been shown by recent studies [8].

Our system evaluates all jobs that are performed by the AGVs, to understand the origins of danger signals and to prevent dangerous jobs from execution. Our system is also designed to minimize the amount of false danger detections, to allow legitimate jobs to be executed uninterrupted. The high-level architecture is shown in Fig. 1. Main modules include the Command and Control that issues job requests; the AGV Controller that generates low-level action plans and chooses appropriate AGVs for execution; the self-learning Danger Detection Module (DDM) that verifies jobs based on their anomaly and danger levels; and the Facility Monitor that alerts the DDM of any independently detected dangers.

The Command and Control (C&C) module oversees the whole manufacturing process. It provides dashboards and control to human operators, but is also

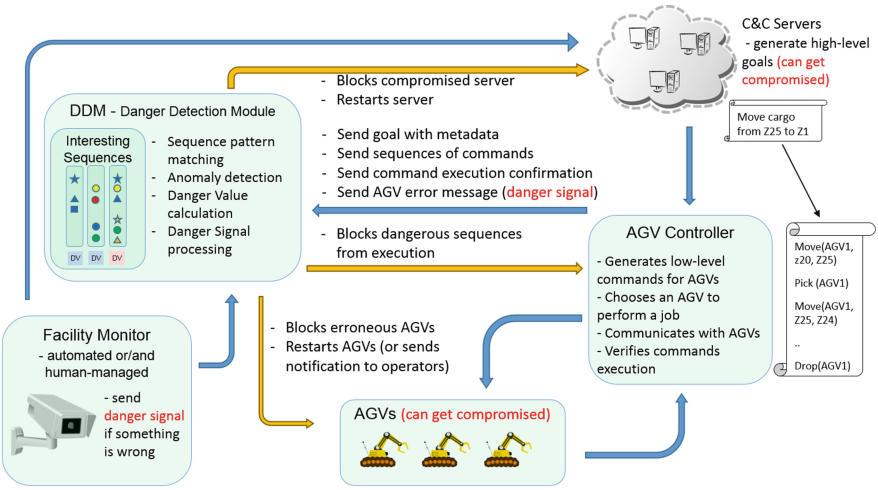


Fig. 1. System Architecture

able to automatically generate high-level requests, such as "Deliver item Omega from collection point A to delivery point B" to support a normal manufacturing process. Requests are passed to the appropriate AGV central controller (AGVC). The AGVC generates a job plan formed from a sequence of AGV atomic actions together with a choice of AGV that should satisfy the original request. This job is then sent for verification to the DDM. If the job is approved by the DDM, then the AGVC executes it by assigning the sequence to the associated AGV.

The Facility Monitor is an independent verification system that monitors the factory floor and raises an alert (called a *danger signal*) if anything goes wrong. Any event that happens unexpectedly, any discrepancy between the expected and observed states of the system causes such an alert, albeit with different severity. Examples of such events include a robot performing an action different from the one on its action queue, a robot not responding to commands, cargo being taken to the position that is not on delivery positions list, etc.

The Danger Detection Module contains a history of jobs and performs similarity detection and clustering of every new job request, to calculate its anomaly score. When a danger signal is observed, the DDM checks current and previously finished jobs to find those that are likely to be the cause of the signal. Anomalous jobs are regarded as much more likely to be the cause, but all jobs are checked. Every danger signal increases the threat level of a job. If the threat threshold is reached, the DDM terminates the job by sending the corresponding command to the AGVC, and rejects similar jobs in the future. If a subsequence of actions within a job is identified as being the cause, the DDM rejects only this subsequence, not the whole job. The AGVC takes this into consideration, and, if possible, generates and resubmits a new plan of actions in order to satisfy the C&C request. On receipt of a sequence from the AGVC, an AGV executes it

and reports back with its final status. Importantly, during movement, the AGV sends updates of its position and status to the AGVC, in the event that the controller needs to modify the remaining elements of the plan for that particular task. All relevant details are also sent to the DDM.

3 Danger Detection Module

The Danger Detection Module can be regarded as a police of the manufacturing facility, in that its main goals are to monitor and verify the safety of all factory operations; find the jobs that create problems; be able to stop them and prevent them from creating problems again. In terms of a robotic manufacturing facility, the DDM should be able to: (1) monitor jobs that are being performed on the facility premises in real-time; (2) have information about how anomalous or potentially dangerous these jobs are; (3) collect information about newly detected dangers in the facility and correlate it with active jobs, potentially finding the cause of danger; (4) if the cause of danger is found, raise an alarm in order to stop the job and forbid the execution of other similar jobs.

The internal architecture of the DDM is shown in Fig. 2. Initially the DDM populates the Knowledge Base with a historical dataset of previous jobs, finding clusters of similar jobs and calculating their parameters, such as frequency rate, anomaly and danger scores. During production, the DDM monitors active jobs, keeping track of all jobs that are currently in progress, their state of execution, i.e. which actions were already performed, and which are planned, etc. The DDM also performs real-time job verification for every new job request that the AGVC creates. This includes finding similar job clusters, calculation the anomaly score of a job, checking previous danger signals of similar jobs, and deciding if a job is a normal one or a dangerous one and should be rejected. If

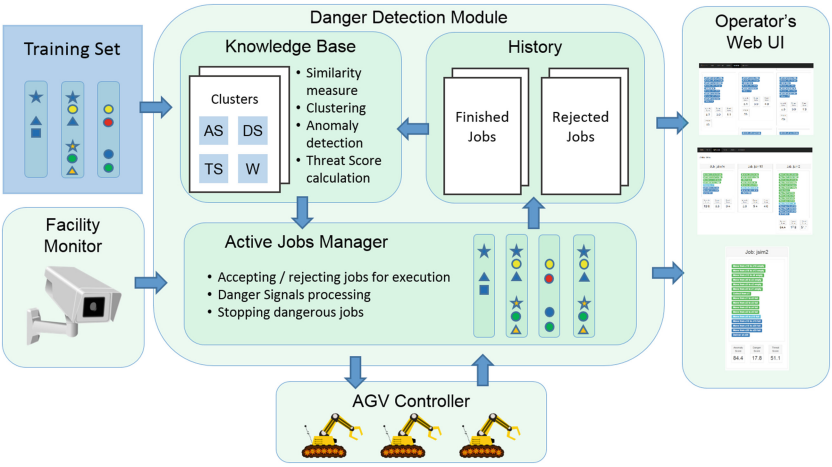


Fig. 2. Internal DDM architecture.

any danger signal arrives, the DDM correlates active jobs with danger signals, deciding on which jobs may be the cause of the signal, and if the danger is severe enough for the job to be immediately stopped. The Knowledge Base is constantly updates with recent data of executed jobs, danger signals, etc., so the algorithm keeps learning and adapting to changing conditions. Finally, the DDM has the Operator's UI that gives capability to human operators to control the system and its decisions.

3.1 Knowledge Base

The Knowledge Base contains a dataset of jobs that correspond to a normal activity of a factory, and performs clustering and anomaly detection, as well as storing the information about the danger score of clusters. Initially, a dataset of historical jobs is used to train the system, to create clusters of similar jobs, and understand their frequency rates. As soon as an active job is finished, for any reason (successful execution or stopping due to danger signals), it is also submitted to the Knowledge Base, in order to update the danger detection dataset. The job that was rejected before being started is not submitted to the dataset.

Similarity Measure. In order to perform job clustering it is necessary to have a measure of similarity between two different jobs.

There are several existing ways to calculate similarity for sequences. Among the most commonly used ones, the Levenshtein distance, the Jaccard similarity, and the longest similar subsequence can all be used in the DDM as a measure of similarity between two sequences of events. The Levenshtein distance (also called "the minimum edit distance") is calculated as the minimum number of atomic operations needed to be performed on an entity in order to transform it into the other entity. The longest similar subsequence metric can be useful in some settings, where the order of actions is very strict and limited, but is weak in the general case, because the small changes in the middle of a sequence will severely lower the total similarity score. We opted to use the Jaccard similarity because it is one of the most general similarity metrics that is applicable to sequences. The Jaccard metric allows to have variations in any part of a sequence, unlike the longest subsequence metric, but can be calculated more efficiently than the Levenshtein distance. The Jaccard similarity is usually used to define the similarity of two sets. In a general case, it is defined as the intersection of two sets divided by the union of two sets: $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$. The similarity metric shows the percentage of items that are the same in two sets to all items in both sets. It returns 1, when both sets are the same, and 0, when there is not a single common item between them. When the Jaccard similarity is used to define the similarity of sequences of variable length with changing token order, the sequence should be transformed into a set [6]. This is commonly done by transforming a sequence into a set of *k-shingles* or *k-grams*. A *k-shingle* is any set of continuous tokens of a sequence. For example, for the sequence "*AirbusGroup*" and letters taken as tokens, a set of all

3-shingles is {"*Air*","*irb*","*rbu*","*bus*","*usG*","*sGr*","*Gro*","*rou*","*oup*"}. Splitting the sequence on k-shingles in order to apply set similarity measures has a number of useful properties, including the ability to cope with small insertions or changes of symbols in random places of a sequence. It is also easy to extend the notion to find a mutual similarity of more than two entities at the same time.

One more important advantage of Jaccard similarity is that it splits the sequence onto subsequences, so it is possible to re-use them to calculate the similarity and danger score of subsequences as well as full sequences. This is very helpful when trying to find the most dangerous subsequence within a sequence. With Levenshtein distance every subsequence would have to be analysed separately, therefore decreasing the total performance of the algorithm.

Clustering. The next step after the similarity between any two entities can be found, is to cluster the set of entities (jobs, or event sequences, in our case) into groups with similar objects. Ideally, each cluster should contain a single type of a job, including small variations that a job can have in its events.

In the DDM we use distance-based hierarchical clustering, with the usage of mean points as representatives of a cluster. We define *a representative sequence* of a cluster as a mean point, i.e. a point that has the maximum similarity to all other points of the cluster $argmax_{c \in C}(\sum_{i \in C} sim(c,i) * W_i)$. For distance-based clustering there is a threshold $MINSIM$, and we require the similarity of all points within a cluster with its mean point to have at least this amount of similarity: $\forall i \in C : sim(c,i) \geq MINSIM$.

Clustering happens sequentially, i.e. we regard one point at a time and add it to the closest cluster (recalculating the mean point if necessary) or create a new cluster if no sufficiently similar cluster is found. The process is the same for initial training and for the production phase, when finished jobs are added one by one. If recalculation of the mean point leads to the cluster no longer satisfying distance requirements, we split the cluster into several smaller ones.

Cluster Parameters. After we have obtained job clusters, we can calculate the parameters of any cluster or any job. For every cluster we calculate the frequency rate (or weight), the anomaly score, the danger score, and the final threat score. In order to calculate the anomaly score of a job, we find the cluster that it belongs to (or create a new one if there is no cluster that is sufficiently similar to a job), and use the anomaly score of this cluster. Conventionally, the algorithms with similar functionality are called 'anomaly detection'. However, here we talk about 'anomaly score calculation', due to the fact that we are not interested in boolean classification of job instances as anomaly vs. non-anomaly, but rather in a quantifiable score of how anomalous the job is.

Weight (W) or **Frequency Rate (FR)** values show, how common the jobs of this cluster are. The weight or the frequency rate can be used interchangeably, with only a small difference in calculation formulas. The weight shows the absolute amount of times the jobs of this cluster were seen within regarded timeframe or within a training set. Frequency rate shows the percentage of times the

jobs are a part of this cluster in comparison to the total number of jobs. As can be easily seen, frequency rate can be obtained by dividing the weight of the cluster by the total weight of all clusters: $FR_i = W_i / \sum_{c \in C} W_c$. Frequency rate is a slightly more adaptable value than weight, when the total amount of jobs over time can vary. However, in certain situations weight can be more preferable, for example, if the total number of jobs within our timeframe is small, and we want to limit the absolute number of jobs for a job to be regarded as non-anomalous.

Anomaly Score (AS) simply represents how anomalous is the job or the cluster. The score is always in the range between and including 0 and 1, where 1 represents that such a job is an absolute anomaly and has never before been seen in the training set, and 0 represents that a job is completely common. From a naive point of view, the AS of a cluster can be seen as being fully dependent on the frequency rate of a cluster, i.e. the higher its frequency, the lower the anomaly score. However, while the frequency rate of a cluster is indeed an important factor in determining the AS, it is not the only factor, as similarity to other clusters and their frequency rates should also be taken into account. For example, two cluster with the same frequency rate will receive different anomaly scores, if the first one has subsequences that are similar to other clusters, and the second one has completely unique sequences. It can be the case that a cluster with lower frequency rate will receive a lower anomaly score, if it has many similar neighbouring clusters that are sufficiently frequent themselves.

Another question is which clusters to regard as completely non-anomalous ($AS = 0.0$). We introduce a frequency rate threshold. It should be chosen to cover the least frequent "normal" job. E.g. if there are three clusters that represent normal activities, one with frequency rate of 0.5, another one with 0.2, and the third one with 0.3, the threshold should be chosen as 0.2. It is also wise to lower the threshold a bit more (e.g. by 10–15 %) to allow for random variations in actual real-time frequencies, therefore finally keeping it at around 0.18. We normalize the frequency rate to obtain the percentage of FR below threshold. The normalized frequency rate (NFR) can be calculated irrespectively of whether original values are represented as absolute weights (W) or as relative frequency rates (FR), however, the threshold should be given in the same units. In case the weight W_i of a cluster is given, the threshold should be given as maximum weight MW, and the normalized frequency rate is calculated as

$$NFR_i = 1 - \frac{max(MW - W_i, 0)}{MW}$$

In case the frequency rate FR_i of a cluster is given, the threshold should be given as maximum frequency MF, and the normalized frequency rate is calculated as

$$NFR_i = 1 - \frac{max(MF - FR_i, 0)}{MF}$$

NFR represents only the frequency of sequences from the cluster itself. However, when calculating the anomaly score we also want to take into account total occurrences of similar subsequences, even when these subsequences are part of

sequences in other clusters. The rate we take from other clusters should be reduced proportionally to the similarity between these clusters. Therefore we introduce extended normalized frequency rate $ENFR_c(C)$ that is calculated for a cluster given a set of clusters for comparison:

$$ENFR_c(RC) = \begin{cases} (1 - Sim(x,c) * NRF_x) * ENFR_c(RC \backslash x) + \\ \quad Sim(x,c) * NRF_x, & \text{for any } x \in RC \\ 0, & \text{if } RC = \emptyset \end{cases}$$

Using the ENFR, we calculate the anomaly score:

$$AS_c = 1 - ENFR_c(C)$$

Note, that it is not necessary to regard the NFR_c of a cluster c separately, if the cluster itself is included into the set C. Because $Sim(c,c) = 1$, the frequency rate of the cluster itself will be taken fully during the calculations.

In practice, the anomaly score of 1.0 cannot be obtained during training phase, because during training phase a sequence is immediately added to the set of sequences, therefore it has some non-zero weight even when seen for the first time. However, during the actual monitoring phase, when a new job is sent to the DDM for verification, it can have anomaly score equal to 1.0. This can be obtained if the new job is not only seen for the first time, but also does not contain any subsequences that were seen previously. Once the fully anomalous job is executed and completed, it will be added to the dataset, and "learned" by the DDM. Therefore the anomaly score of a similar job next time will be lower. The anomaly score of 0.0 can be obtained during training or verification phases for all clusters that have normalized frequency rate of 1.0.

Danger Score (DS) is the metric that shows, how many danger signals were detected during the execution of these jobs, how severe they were, and how likely it is that they were caused by the jobs from the cluster, and not some other jobs. A danger score of a cluster increases when active jobs from this cluster are associated with environmental danger signals. Each danger signal has a severity value. This value gets distributed among active jobs that may be responsible for the signal, depending on the type of the signal and job parameters, such as their anomaly score and previously associated danger signals. More on the distribution of the danger signal score is explained in Sect. 3.2.

A danger signal chunk ds_{t+1} that gets assigned to a cluster i during step $t+1$ is always in the range of 0.0–1.0. The increase of the total danger score happens according to the following formula:

$$DS_{t+1}^{(i)} = DS_t^{(i)} + (1 - DS_t^{(i)}) * ds_{t+1}$$

Threat Score (TS) represents the total potential perceived threat of executing a sequence of a cluster. It is a combination of how anomalous the sequence is (AS) and how often and severe danger signals related to the sequence are (DS).

In principle, threat score of a cluster can be any function of its anomaly score and danger score, $TS_c = F(AS_c, DS_c)$, as long as the following conditions hold:

1. TS_c takes values in the range of 0.0–1.0;
2. TS_c increases monotonously when AS_c increases;
3. TS_c increases monotonously when DS_c increases.

Currently we use linear formula $TS_c = \alpha * AS_c + (1 - \alpha) * DS_c$. However, other functions can be taken into consideration in the future.

3.2 Danger Signals

Danger signals can include anything that happens in the environment not accordingly to expectations or that harms the environment or the system. The origins and amount of information given by the signals can differ. Signals can be created by AGVs themselves (any error during execution can be a reason for such a signal) or the AGV controller (e.g. if an AGV stops responding). But one of the most reliable methods to obtain the information about the dangerous activity is an independent Facility Monitor. It should have the information about the goals of the jobs, their preconditions and effects, and general rules of the environment (e.g. "location Z25 is a cargo delivery point"). For some signal it is possible to pinpoint exactly the location and the cause of it, while for others such information may be unavailable, requiring the DDM to check all possible jobs in progress. When a danger signal is detected, it gets assigned to active jobs accordingly to internal calculation of probability of this particular job being the cause of the danger. This depends on a type of the danger signal and on an anomaly score of active jobs.

Danger signals have two parameters: type and severity. Severity is a numerical value that represents the expected harmful potential of the signal. For critical danger signals with high severity, a single signal is enough to cause a job to be stopped. For minor danger signals, only sustained repetition of them for the same jobs again and again will cause these jobs to be regarded as threatening. Possible danger types that we expect for the robotic manufacturing facility include: a wrong action performed by a robot, a robot stops responding, or responds erratically, goods disappearing from collection or delivery points; a foreign object is detected on one of the locations, or a general danger alarm of unknown origin.

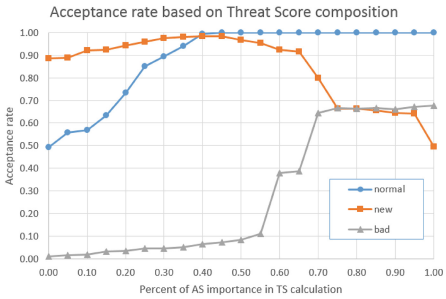
Every danger signal gets distributed among jobs that may have caused it. The exact distribution function varies depending on the type of the signal. I.e. if a robot stops responding, the job that it was executing at the moment is regarded as the main candidate. For a general danger alarm all jobs are regarded as candidates. For a signal that is detected for a particular location, robots are regarded as likely candidates in proportion to their distance to this location. For all danger types, the distribution is further modified by the threat score of potential candidates. A job with higher anomaly score and more danger signals associated with it previously, will receive a higher chunk of the danger signal.

3.3 Active Jobs Manager

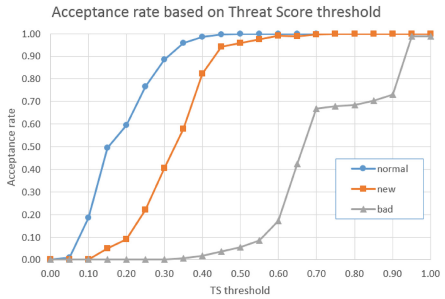
Main goals of the Active Jobs Manager (AJM) is to verify jobs, to keep the record of jobs that are currently being executed and to track their successful execution. In the presence of danger signals, the task of the AJM is to find out which active jobs may be associated with this signal, and determine if their threat level is enough to issue a command to stop them and forbid similar jobs in the future. The AJM works in real-time. When a new job is created by the AGVC, it is first sent for verification to the AJM. The AJM either approves or rejects it, based on its knowledge of previous harmful activities. If a job is rejected, the dangerous subsequence within the job is sent back as a reason for rejection. The AGVC can then try to recreate the job, to fulfill the goal using a different plan of execution that avoids this subsequence. If a job is approved, the DDM adds it to the list of active jobs. When any job event is completed by an AGV, the AGVC sends a confirmation to the DDM, so the DDM always knows at which state the job is. If a danger signal is received by the DDM, the AJM applies it to related active jobs accordingly to rules of danger score distribution. After this is done, a new verification is done to affected jobs, in order to decide if they should be stopped. If this is the case, the DDM will send a stop command to the AGV Controller.

4 Evaluation

We performed a number of experiments to assess the approach in terms of danger detection. The experiments are based on a simulated factory floor with three AGVs operating simultaneously. The factory floor has several collection and delivery points. As a first step, we create a training set by generating two thousand jobs to collect at a random collection point and deliver to a random delivery point. This dataset represents a set of "normal activities". Then we run the system in "production mode", where we generate one thousand jobs in a similar fashion, but with two additions. The first addition is the addition of new legitimate behaviors. This is done by defining a new collection point or a new delivery point. Such activities are expected to have high anomaly score, due to a robot performing previously unseen sequences of actions, but lead to intended results that can be verified by the Facility Monitor. The second addition to jobs is the addition of "bad jobs". We assume that an instance of the C&C got compromised, and sends a request to a robot to deliver goods to a wrong place that is not intended, where the goods can be collected by attackers. Due to the fact that goods disappear from the factory floor without arriving to the intended destination, the Facility Monitor raises an alarm and creates a danger signal as soon as it notices that goods have disappeared. However, since the DDM does not know which new locations are intended and which are not, it has to reason as described in Sect. 3, by distributing the Danger Signal among related jobs. If the threat score (TS) of a job reaches a predefined threshold, the job is stopped and is marked as dangerous. Further similar jobs are rejected. For every set of parameters we create one hundred randomized system runs, each consisting of 2000 jobs training set and 1000 jobs verification set, and take the average results.



(a) Acceptance rate based on α in $TS_c = \alpha * AS_c + (1 - \alpha) * DS_c$.



(b) Acceptance rate based on TS threshold for job rejection.

Fig. 3. Acceptance rates depending on parameters

The decisions of the DDM depend on how the threat score is calculated and treated. Therefore, as the first experiment we look at the parameter α in the equation $TS_c = \alpha * AS_c + (1-\alpha) * DS_c$. We vary α in the range of 0.0–1.0 with 0.05 step. For the purpose of this experiment we fixed the TS threshold at 0.55, i.e. any job with calculated TS higher than 0.55 is immediately stopped or is rejected from the beginning. We calculate the acceptance percentage for "normal jobs" (i.e. jobs that have collection and delivery points available in the training set), for "new jobs" (i.e. legitimate jobs, but with previously unseen collection or delivery points), and for "bad jobs" (i.e. jobs with a previously unseen and wrong delivery zone that cause disappearance of goods). Results are shown in Fig. 3a. Note that with $\alpha = 0.0$ anomaly score plays no role in the decision whatsoever, only danger signals matter. Danger signals appear closely associated with dangerous jobs, therefore most "bad" dangerous jobs are rejected, with acceptance rate staying below 0.1 up for all $\alpha < 0.55$. However, because we do not take any anomaly score into account, and only look at similarity of jobs to the ones associated with danger signals, normal jobs become associated with danger signals as often as new or bad jobs. This leads to a high rate of rejection for normal jobs ("false positives"). With increasing α, the danger signal distribution starts to take into account the anomaly value, therefore normal jobs become less and less likely to be associated with danger signals, and their acceptance rate increases rapidly, reaching values close to 1.0 at $\alpha > 0.4$. We see the best results with α in the range of 0.4–0.5, with very high acceptance of normal and new legitimate jobs, but very low acceptance of bad jobs (due to remaining importance of danger signals). However, when α increases past 0.5, we see a dramatic drop in acceptance of new jobs (due to the fact that the anomaly score is now very important, but the existence or absence of danger signals is not important). Bad jobs become more accepted as well, reaching the same percentage of acceptance as new jobs for $\alpha > 0.75$. This is due to anomaly values of new legitimate jobs and new bad jobs being on the same level, and lack of importance of danger signals to discriminate legitimate and bad behavior.

The TS threshold (TST) is also an important parameter, therefore as our second experiment we vary the TST in the range of 0.0–1.0, but now with fixed $\alpha = 0.5$. The results can be seen in Fig. 3b. We can see that with $TST = 0.0$ all jobs get rejected immediately, but with increasing TST the "normal" jobs increase their acceptance rate rapidly, with about 60 % of normal jobs being accepted with TST as low as 0.2. This is due to threat score for normal jobs being usually very low, due to low anomaly score as well as low to none relation to danger signals. However, with low TST new legitimate jobs are largely rejected, due to anomaly score alone being enough to breach the threshold. Values around 0.4–0.5 produce the best results: most normal and new jobs have the threat score lower than this threshold, and are therefore accepted, but the combination of anomaly score and related danger score leads to most bad jobs breaching the threshold and being successfully identified and rejected. With increasing the TST beyond 0.5 results worsen, as bad jobs become more and more accepted as well. It should be noted, however, that even with $TST = 0.9$ about 30 % of bad jobs are still identified and rejected.

5 Conclusions

This paper presents an approach for intelligent detection of and response to threatening activities in Cyber-Physical Systems. The system is able to recognize anomalous activities and environmental dangerous events, and relate them in order to understand, which jobs may have been the cause of the danger. Such jobs can be stopped and prevented in the future. The system, presented here, demonstrates the concept of an intelligent self-aware manufacturing facility.

It is important to mention that the Danger Detection Module is implemented in a domain-independent way. Detecting anomalies and dangerous activities in sequences of events is a general topic that can be applied to other settings as well as in safeguarding robotic manufacturing facilities. The Danger Detection Module can be applicable in such settings as incident detection in network traffic, analysis of system calls, safety of smart homes, etc.

References

1. Cani, A., Gaudesi, M., Sanchez, E., Squillero, G., Tonda, A.: Towards automated malware creation: code generation and code integration. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing, pp. 157–160. ACM (2014)
2. Forrest, S., Perelson, A.S., Allen, L., Cherukuri, R.: Self-nonself discrimination in a computer. In: 2012 IEEE Symposium on Security and Privacy, p. 202. IEEE Computer Society (1994)
3. fr Sicherheit in der Informationstechnik (BSI), B.: Die lage der it-sicherheitin deutschland (2014). https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2014.pdf?__blob=publicationFile
4. Kim, J., Bentley, P.J., Aickelin, U., Greensmith, J., Tedesco, G., Twycross, J.: Immune system approaches to intrusion detection-a review. Natural Comput. **6**(4), 413–466 (2007)

5. Langner, R.: Stuxnet: dissecting a cyberwarfare weapon. Secur. Priv. IEEE **9**(3), 49–51 (2011)
6. Manber, U., et al.: Finding similar files in a large file system. In: Usenix Winter, vol. 94, pp. 1–10 (1994)
7. Matzinger, P.: Tolerance, danger, and the extended family. Annu. Rev. Immunol. **12**(1), 991–1045 (1994)
8. Petit, J., Shladover, S.: Potential cyberattacks on automated vehicles. IEEE Trans. Intell. Transp. Syst. **16**(2), 546–556 (2015)