# Malicious Human Behaviour in Information System Security: Contribution to a Threat Model for Event Detection Algorithms

Olivier de Casanove[1,2(✉)] and Florence Sèdes[1,2]

[1] Université Toulouse III - Paul Sabatier, 118 route de Narbonnes, 31062 CEDEX 9 Toulouse, France
[2] Institut de Recherche en Informatique de Toulouse (IRIT), Toulouse, France
{olivier.decasanove,florence.sedes}@irit.fr

**Abstract.** Among the issues the information system security community has to fix, the security of both data and algorithms is a concern. The security of algorithms is dependent on the reliability of the input data. This reliability is questioned, especially when the data is generated by humans (or bots operated by humans), such as in online social networks. Event detection algorithms are an example of technology using this type of data, but the question of the security is not systematically considered in this literature. We propose in this paper a first contribution to a threat model to overcome this problem. This threat model is composed of a description of the subject we are modelling, assumptions made, potential threats and defence strategies. This threat model includes an attack classification and defensive strategies which can be useful for anyone who wants to create a resilient event detection algorithm using online social networks.

**Keywords:** Threat model · Adversarial Learning · Online Social Network · Event Detection · Security

## 1 Introduction

The reliability of the output data of a machine learning algorithm is determined, among other factors, by the reliability of the input data. A small perturbation in the input can result in a misleading output [25]. This perturbation may be due to either data gathering or malicious behaviour. When the input data are generated by human, perturbations due to malicious behaviour cannot be disregarded. Data from online social networks is an example of data generated by human or prone to malicious perturbation.

Online Social Networks (OSN) allow users to exchange short messages and media. From these data published in real time, information can be extracted on events. Atefeh and Khreich [5], in their review on event detection on OSN, stated

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
G.-V. Jourdan et al. (Eds.): FPS 2022, LNCS 13877, pp. 208–220, 2023.
https://doi.org/10.1007/978-3-031-30122-3_13

that "in general, events can be defined as real-world occurrences that unfold over space and time". To detect these events, event detection algorithms can be used. They take as an input the OSN stream of messages and give as an output clusters of messages, where each cluster defines an event. There are many applications, ranging from earthquake detection to musical event detection [5]. The literature on the subject focuses on how to improve the performances of the detection and the question of the detection's security is neglected. Yet, when the input data is made up of messages crafted by unknown users, this subject becomes a concern. If the implicit hypothesis: "Input data are absent from malicious messages crafted in order to disrupt the event detection" may hold in specific contexts, it does not in others. For example, when detecting events related to cybersecurity, the adversaries want their attacks to stay undetected. In this field, it's easy to find papers which do not take into account a potential threat to their detection [14,21]. When this hypothesis is false, we find ourselves in an adversarial learning context and event detection is under many threats.

This paper is a first contribution to a threat model for event detection algorithms on Twitter, but the same threat model could be used for other OSN. According to OWASP (Open Web Application Security Project) [19], a threat model is "a structured representation of all the information that affects the security of an application. In essence, it is a view of the application and its environment through the lens of security. [...] A threat model typically includes:

- Description of the subject to be modelled.
- Assumptions that can be checked or challenged in the future as the threat landscape changes
- Potential threats to the system
- Actions that can be taken to mitigate each threat
- A way of validating the model and threat and verification of success of actions taken."

Our contribution addresses all the points of this definition except for the fifth one, which will be addressed in future works. We believe that this threat model can be used to develop more resilient event detection algorithms and therefore, develop a technology more suited for real-life applications.

In the next section, we briefly discuss the related work. We will use the previous definition of a threat model to structure the rest of our paper. In Sect. 3, we describe the subject we model. In Sect. 4 we provide the assumptions on which our model is based. In Sect. 5 we describe the threats. In Sect. 6 we present defence strategies. In Sect. 7 we present future works and possible extensions to our threat model. Finally, we conclude in Sect. 8.

## 2 Related Work

Adversarial learning is a recent field, yet there is already a good literature on it [25]. Current threat models for machine learning focus on three aspects: attack direction (does the attack happen during the learning phase or the classification

phase?), security violation (which kind of security concept the attack violates, traditionally confidentiality, integrity and availability) and attack specification (is it a targeted attack or not). We will see in Subsect. 5.2 why this threat model is not useful for us, as it is. Adversarial learning specifically applied to OSN has been studied in three different ways. The first one focuses on text processing applications, which is what event detection algorithms are. Alsmadi et. al. [3] review the literature to categorise attacks against text processing applications. All the attacks identified are message-level attacks, while event detection algorithms could also be attacked at the event-level (set of messages); therefore attacks against event detection algorithms are not fully covered in [3]. The second one is more specific, it is about evading spam detection [10]. Cleaning the input data with spam detection is a common and useful preprocessing step for text processing applications, but once again these attacks focus on message-level. For the same reason as in the previous point, this does not fully cover attacks against event detection algorithms. The third one consists in listing the adversaries and threats which can be faced in OSN. For example, Sabottke et al. [22] proposed an event detection algorithm with a list of actors willing to disrupt their algorithm. This is a first step, we used this list as a basis to construct the list of profiles in Subsect. 5.1, but this work needs to be extended into a complete threat model. Finally, the subject of fake news is out of scope because they impact the users. They are not meant to disrupt the operation of a machine learning algorithm.

## 3 Modelling Event Detection Algorithms

Atefeh and Khreich [5] as well as Hasan et. al. [11] reviewed the literature to list the techniques used to detect events on social media. Regardless of the technique used, an event can be formalised with the following definition:

**Definition 1 (Event).** *An event is defined by a tuple of messages related one to another and which are in the same spatial or time window. We note an event $e_k$, where $e_k \in \mathbb{E}$ and $\mathbb{E}$ is the set of all events possible and $k \in \mathbb{N}$ is the unique identifier of the event.*

We define the function $F$, the function which associates to a tuple of messages the corresponding event if the messages actually form an event and $e_0$ otherwise. Here, $e_0$ symbolise the null event, which means that the messages are not related. The set of all messages possible (or in our case all tweets possible) is noted $\mathbb{T}$.

**Definition 2 (Event Detection function).**

$$F : \mathbb{T}_1 \times ... \times \mathbb{T}_n \to \mathbb{E}$$
$$(t_1, ..., t_n) \mapsto \begin{cases} e_k & \text{if } t_1, ..., t_n \text{ form an event} \\ e_0 & \text{else} \end{cases}$$

An attacker can create fake messages thanks to techniques such as Markov Chain or Neural Network. When executed, these algorithms will produce a new fake message contained in a set of messages the algorithm is able to generate. Therefore we can represent the fake message by a random variable.

**Definition 3 (Fake message random variable).** *Let $X$ be a random variable following an unknown distribution over the set $\mathbb{T}$.*

We define a false positive event (FP event), in the adversarial context, as an event which is composed of both legitimate and crafted tweets, but if the legitimate tweets were to be considered alone, no event would be triggered.

**Definition 4 (False Positive Event).**

*Let $(X_i)_{i\in\mathbb{N}^*}$ be independent and identically distributed variables following the same law as $X$, $\forall (t_j)_{j\in\mathbb{N}^*} \in \mathbb{T}$ and $\forall k \neq 0$ if $F(t_1, ..., t_n) = e_0$ and $F(X_1, ..., X_m, t_1, ..., t_n) = e_k$ then $e_k$ is a false positive*

In opposition, we define a true positive event (TP event), in the adversarial context, as a set of messages mainly composed of legitimate messages and recognised as an event.

**Definition 5 (True Positive Event).**

*Let $(X_i)_{i\in\mathbb{N}^*}$ be independent and identically distributed variables following the same law as $X$, $\forall (t_j)_{j\in\mathbb{N}^*} \in \mathbb{T}$ and $\forall k \neq 0$ if $F(t_1, ..., t_n) \neq e_0$ and $F(X_1, ..., X_m, t_1, ..., t_n) = e_k$ then $e_k$ is a true positive*

## 4 Assumptions

As previously said in Sect. 1, a threat model needs assumptions. We identify three assumptions for this threat model to make sense.

**Assumption 1.** *Input data from Twitter, and more generally social networks, contain messages written by malicious users with the objective to deceive event detection algorithms taking this data as input.*

We know that extracted data from Twitter contain spams and other malicious messages like phishing, for example. Those messages have an influence on the quality of the detection of our algorithms. Working in an adversarial context means taking the idea one step further and supposing that malicious users craft messages just to disrupt event detection algorithms.

**Assumption 2.** *Attackers have access to the algorithms, training and test dataset and any other relevant information.*

The datasets used to compare event detection algorithms are public, papers describing how event detection works are also easily accessible; therefore it is safe to assume that the attackers have access to any information related to event detection. It also means that the system is a "grey box" for the attackers, they have at least partial knowledge of how it works. Security by obscurity is not an option here.

**Assumption 3.** *The benefit of disrupting the detection for the attacker is equal to the cost for the defender to see its detection disrupted.*

We make this assumption to model the adversarial context as a zero-sum game. A zero-sum game, in game theory, is a situation where the benefit of a player (i.e. the attacker) is exactly equal to the cost of the other player (i.e. the defender). Interesting properties could be derived from zero-sum game, we will use them in a future work to validate the model. This is a common assumption in adversarial problems [24,28] and in information system security in general [26,27].

## 5 Threats to the System

In our context, a threat is defined as the combination of a malicious actor (the attacker) and a means to disrupt the event detection (the attack). We will detail both the attacker and the attack in the next two subsections.

### 5.1 Attacker Profiles

In a previous paper, we reviewed other contributions [8] and identified three profiles in the literature: trolls, spammers and adversaries. We will summarise these three profiles except for the troll where we can give a better definition than the one originally given. This gives us the following attacker classification:

- Trolls: their objective is to create rumors or make disappear subjects and therefore, events. They target both humans and automatic tools which analyse the news. In the second case, their objective is to create FP events and make TP events disappear.
- Spammers: they publish a lot of messages serving their own interests. They can use buzzwords, keywords or tag people to improve the efficiency of the spamming activity. They do not target our algorithm directly, but their activity creates a lot of noise in the Twitter stream.
- Adversaries: their objective is specifically to attack the event detection algorithm, in every way possible. Their means are diverse, but we can suppose that they have at least partial knowledge of the technology behind event detection since they are directly targeting it.

The definitions of the attackers are centred around the impact he could have. These profiles could be refined with two additional criteria: 1) is the attacker ignorant or knowledgeable of the system? And 2) is the attacker constrained or free of any constraints? Indeed the attacker could have multiple types of constraints, economic or political, for example. Now that we discussed about the profiles of the attackers, let's continue with the type of attacks they can use.

### 5.2 Attacks

In information system security, the CIA model (Confidentiality, Integrity and Availability) is often used [23]. However, this model does not suit our needs well in our adversarial learning context. For example, it does not make sense to defend the confidentiality of the detected events when all of the message composing it are public and messages. We propose instead to use the reliability and validity, which are measurement properties. "A measurement property is a quality aspect of an instrument" [1], i.e. event detection algorithms. "Each measurement property requires its own type of study to assess it" [1].

**Reliability.** The reliability of a test is its ability to stay consistent. In other words, a same input should always give the same output. In our adversarial context, the reliability becomes 1) the ability of the event detection algorithm to detect a same event, both when the input data are not corrupted and when a malicious actor is tampering with the input messages, with no less messages in it and 2) the ability to detect an event, with no more messages in it, when a malicious actor is tampering our data. To measure the impact on consistency, we need to first run our algorithm on a dataset without fake messages and label the messages associated to an event. We run again our algorithm, this time with the fake messages in the dataset and we compare the new clusters to the initial labels. This is a clustering problem, therefore a relevant metric for clustering tasks should be used. Traditional metrics such as recall or precision are not the best option for that. According to their review of different metrics for clustering tasks, Amigo et. al. [4] conclude that the best metrics, in regard of the properties they defined in their paper, are the BCubed recall and the BCubed precision combine into the BCubed F1-Score; therefore we choose these metrics for our problem. They define BCubed recall as"how many items from its category appear in its cluster" [4], which match our first objective for reliability and the define BCubed precision as "how many items in the same cluster belong to its category" [4], which match our second objective. The BCubed precision and the BCubed recall are calculated for every cluster, then "The overall BCubed recall [respectively the BCubed precision] is the averaged BCubed recall [respectively BCubed precision] of all items in the distribution" [4]. These two metrics are then combined to create the BCubed F1-score in the same way that traditional precision and recall are combined to create the F1-score. In conclusion, we will use the BCubed F1-score to measure the reliability.

$$\text{Reliability} = \text{BCubed F1-Score}$$

The exact formula of the BCubed F1-score is complex and would require the introduction of multiple notions. We encourage the interested readers to go read Amigo et. al. [4] if they want to go into BCubed metrics in depth.

**Validity.** The validity of a test is its ability to detect what it pretends to detect. In our case, is the event detection algorithm detecting events and not

just give a random output? The objective of the event detection algorithm is therefore to maximise TP events and minimise FP events. The precision metric increases when the number of TP increases and decreases when the number of FP increases; therefore we will use the precision to measure the validity.

$$\text{Validity} = \text{Precision} = \frac{TP}{TP + FP}$$

**Attacks Classification.** After studying event detection algorithms, we identify eight attacks. These attacks are described in terms of their impacts on the reliability and the validity of the detection. They are classified in three categories: event creation, event dispersion and event modification. These categories gather the attacks which use the same means, but they don't always have the same goal. We summarise the attacks in Table 1.

*Event creation.* The attacker triggers an event detection, which increases the FP events and therefore impact the validity. The attacker uses a tool to procedurally generate fake tweets. Those messages are then injected in the Twitter stream. We identify three attacks in this category:

- Craft: fake tweets are created. Those messages are close enough for the event detection algorithm to consider them as related but does not necessarily make sense for a human. Those messages trigger a detection.
- Message expansion: real tweets, not related to any event, in association with malicious tweets trigger an event. This attack also impacts the reliability since a legitimate message, not related to any event, becomes related to an event.
- Replay: A TP event is replayed, entirely or partially, at a time where the event doesn't make sense.

*Event dispersion.* The objective is to inject enough malicious messages during a small lapse of time so the legitimate tweets appear too far from one another in the Twitter stream for the event to be detected. Three attacks exist in this category:

- Fragmentation: an event is split in two or more subgroups of tweets, resulting in detection of multiple events when they are the same. One TP events become many TP events under attack; therefore the reliability is impacted.
- Cancellation: an event doesn't trigger a detection when it should. The tweets are so split by the malicious messages that they aren't recognised as an event anymore. This attack decreases the number of TP events and transforms a TP event in nothing, therefore both the validity and the reliability are impacted.
- Deterioration: the number of tweets in an event decreases when under attack. This is a mix case between fragmentation and cancellation. The first or last messages are too far to be associated with the event, but they are still enough messages to trigger a detection. This is an inconsistency under attack; therefore it impacts reliability.

**Table 1.** Attacks against event detection algorithms

| Name | Description | Impact on Reliability | Impact on validity |
|---|---|---|---|
| *Event Creation* | | | |
| Craft | A collection of fake tweets triggered an event | NO | YES |
| Message Expansion | A collection of fake and real tweets, which wouldn't have triggered an event otherwise, triggered an event | YES | YES |
| Replay | A true event is replayed a second time by the attacker | NO | YES |
| *Event Dispersion* | | | |
| Fragmentation | An event triggered multiple detection due to spam activities | YES | NO |
| Cancellation | An event doesn't trigger detection due to spam activities | YES | YES |
| Deterioration | The number of tweets related to an event is less than expected due to spam activities | YES | NO |
| *Event Modification* | | | |
| Drift | The attacker change the event keywords or event | YES | NO |
| Merge | Messages from an event start to aggregate to another event | YES | NO |

*Event modification.* The attacker generates malicious tweets which seem related to one another by the event detection algorithm. As for event creation, the messages are generated procedurally.

- Drift: the attacker creates malicious tweets which aggregate on a TP event. The objective is to change the event keywords or subject. It creates an inconsistency; therefore the reliability is impacted.
- Merge: the attacker changes the event keywords or subject so another event messages start to aggregate on the first event. For this attack to be successful, the attacker needs to know the subject of two different events. It is safe to assume that if the attacker knows this, then both events already have been detected by our algorithm. Therefore it only creates an inconsistency on the number of messages aggregated to each event, and not in the number of TP events detected. The reliability is impacted.

## 6 Defence Strategies

The defender can protect the detection by adding filters at two different levels. The first one is at the level of the tweets, where tweets which seem malicious are filtered. The second level is at the level of the cluster, where TP events are

distinguished from FP events. We define the filter function $h$, the function which associates, to each set of messages recognised as an event, 0 if the set of messages does not satisfy the constraints or a unique value otherwise.

**Definition 6. (Filter Function).**

$$h : \mathbb{E} \to \mathbb{E}$$
$$e_k \mapsto \begin{cases} e_k & \text{if } e_k \text{ satisfies the filters} \\ e_0 & \text{else} \end{cases}$$

With this new element in mind, we redefined TP and FP event as follows:

**Definition 7. (True Positive Event).**

*Let* $(X_i)_{i\in\mathbb{N}^*}$ *be independent and identically distributed variables following the same law as* $X$, $\forall (t_j)_{j\in\mathbb{N}^*} \in \mathbb{T}$ *and* $\forall k \neq 0$ *if* $F(t_1, ..., t_n) \neq e_0$ *and* $(h \circ F)(X_1, ..., X_m, t_1, ..., t_n) = e_k$ *then* $e_k$ *is a true positive*

**Definition 8. (False Positive Event).**

*Let* $(X_i)_{i\in\mathbb{N}^*}$ *be independent and identically distributed variables following the same law as* $X$, $\forall (t_j)_{j\in\mathbb{N}^*} \in \mathbb{T}$ *and* $\forall k \neq 0$ *if* $F(t_1, ..., t_n) = e_0$ *and* $(h \circ F)(X_1, ..., X_m, t_1, ..., t_n) = e_k$ *then* $e_k$ *is a false positive*

We will now discuss what the defence strategies are. Table 2 summarises which defence strategies mitigate which attacks.

### 6.1 Filtering Messages

The objective of a spam filter is to distinguish fake users, spams and spammers from legitimate tweets and users [2]. A spam filter can be made on the content of the tweets, the characteristics of the tweets, the users behind the tweets or the relationships in the OSN of the users behind the tweets [2]. All these solutions are machine learning solutions; therefore we introduce a new level of adversarial learning. However, the problem of adversarial learning for spam detection has already been discussed by [6,7,9,13]. Generating fake messages that can fool the spam filter increases the cost of the attack. Therefore, this strategy is effective against every attack which needs to create fake tweets. It is especially effective against dispersion attacks since those attacks are based on flooding and flooding are easily detected by spam filters. Finally, spam detection based on user features is effective against replay attacks because it means that the accounts replaying the events should avoid spam detection; therefore it increases the cost of the attack.

### 6.2 Filtering Clusters

TP and FP events have different characteristics. Setting thresholds for these characteristics is a way to differentiate TP from FP events. These thresholds are used as filters to discard FP events. We identified five metrics in the literature on which events can be filtered:

- Word entropy: The entropy of a cluster was introduced by [20]. The formula (1) is used where X is a random variable and $P(X_i)$ is the probability to draw a specific word out of all the words of the cluster. A cluster with a very low word entropy is probably composed of very similar crafted messages.
- User diversity: The formula (1) is applied but instead of applying it to words, it is applied to users in the cluster. We have X a random variable and $P(X_i)$ the probability to draw a specific user out of all the users of a cluster. *User diversity* in a cluster was introduced by [15]. This metric is particularly interesting because accounts are the most difficult thing to fake as an attacker. *User diversity* is one of the rare defence measures against event replay. The attacker can replay the exact same tweets but not the exact same author.
- Least Common Subsequence (LCS): Hasan et al. in [12] use a filtering method based on the LCS at word-level. The idea, based on empirical evidence they found, is that cluster of newsworthy events will have a higher LCS than non-newsworthy events. In their paper, the authors fixed an LCS threshold under which an event is discarded. It may help to identify drifted and merged events since the first and last messages of these events are likely to be very different.
- Named entity recognition: This technique is introduced by [18] as a way to pre-select tweets with significant improvement in the final result. The argument behind this constraint is that a tweet without a named entity does not provide any information and is therefore useless.
- Event size: Intuitively a cluster of fewer than 3 tweets cannot be considered as an event. However, finding an exact event size threshold separating meaningful events from similar but not related messages is impossible. Event size should be considered as a hyperparameter of our model to help us drop FP events.

$$H(X) = -\sum_{i}^{n} P(X_i) log_b P(X_i) \tag{1}$$

Some of the filter proposed are easy to bypass. For example, attackers can automatically add a random named entity in their fake tweets. We should keep in mind that, for the attacker, every attack is a trade-off between the costs and benefits of the attacks. Therefore, every defence strategy increasing the cost of the attack is worthwhile.

### 6.3 Other Strategy

Defragmentation is a process where events are reviewed to check if two detected events are in fact only one. Some event detection algorithms are prone to

fragmentation [5]. Our context adds another interest to defragmentation: the resilience to the event splitting attack. We found one utilisation of defragmentation in [12].

**Table 2.** Defence strategies

| Attack | Defence strategy |
|---|---|
| Tweets Expansion | Spam filters, Cluster filters |
| Event Crafting | Spam filters, Cluster filters |
| Event Replay | Spam filters, User diversity filter |
| Event Fragmentation | Spam filters, Defragmentation |
| Event Cancellation | Spam filters |
| Event Deterioration | Spam filters |
| Event Drifting | Spam filters, LCS filter |
| Event Merging | Spam filters, LCS filter |

## 7 Future Works

We seek in future works to validate the model with a mathematical proof of how this formalisation is relevant. Thanks to our definition of TP and FP events, the hypothesis 3 and game theory theorems, we can prove that it exists a point where neither the attacker nor the defender will have interest into changing their strategies; therefore we avoid the pitfall of the Red Queen hypothesis [17]. The Red Queen hypothesis, in cybersecurity, is the hypothesis that there is a form of coevolution between attackers and defenders. Attackers develop their offensive strategies and the defenders develop countermeasures, which will lead to the attackers to change their strategies and so on.

On another note, we would like to develop a solution which could emulate the attacks in 1. We will need for that a public dataset for event detection [16] and a text generator able to generate credible messages and credible set of messages to form an event. The solution would automatically insert the fake messages in the dataset and the event detection algorithms would be tested on this new dataset. The resilience of the event detection algorithms would be measured thanks to reliability and validity. This future work can help to test the resilience of event detection algorithms.

## 8 Conclusion

In this paper we proposed a first contribution to a threat model for event detection. We define the situation we are modelling, assumptions that were made, the attackers' profile, possible attacks and defence strategies. In future works we will

propose a way to validate our model. This threat model includes an attack classification and defensive strategies. This work is dedicated to help future event detection algorithms to be more resilient against adversarial attacks and therefore, develop a technology more suited for real-life applications. This threat model is especially useful when the event detection algorithms detect events related to any subject where an adversary can be found.

## References


1. COSMIN Taxonomy of Measurement Properties • COSMIN. https://www.cosmin.nl/tools/cosmin-taxonomy-measurement-properties/
2. Abkenar, S.B., Kashani, M.H., Akbari, M., Mahdipour, E.: Twitter Spam Detection: A Systematic Review. arXiv:2011.14754 [cs] (2020). version: 2
3. Alsmadi, I., et al.: Adversarial Attacks and Defenses for Social Network Text Processing Applications: Techniques, Challenges and Future Research Directions. arXiv:2110.13980 [cs] (2021). http://arxiv.org/abs/2110.13980
4. Amigó, E., Gonzalo, J., Artiles, J., Verdejo, F.: A comparison of extrinsic clustering evaluation metrics based on formal constraints. Inf. Retr. **12**(4), 461–486 (2009). https://doi.org/10.1007/s10791-008-9066-8
5. Atefeh, F., Khreich, W.: A Survey of techniques for event detection in Twitter. Comput. Intell. **31**(1), 132–164 (2015). https://doi.org/10.1111/coin.12017
6. Biggio, B., Fumera, G., Roli, F.: Design of robust classifiers for adversarial environments. In: 2011 IEEE International Conference on Systems, Man, and Cybernetics, pp. 977–982 (2011). https://doi.org/10.1109/ICSMC.2011.6083796, ISSN: 1062-922X
7. Brückner, M., Kanzow, C., Scheffer, T.: Static prediction games for adversarial learning problems. J. Mach. Lear. Res. **13**(1), 2617–2654 (2012)
8. de Casanove, O., Sèdes, F.: Apprentissage adverse et algorithmes de détection d'évènements : une première typologie. In: Rendez-vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information (RESSI 2022) (2022). https://hal.archives-ouvertes.fr/hal-03668829, poster
9. Chan, P.P.K., Yang, C., Yeung, D.S., Ng, W.W.Y.: Spam filtering for short messages in adversarial environment. Neurocomputing **155**, 167–176 (2015). https://doi.org/10.1016/j.neucom.2014.12.034
10. Duddu, V.: A survey of adversarial machine learning in cyber warfare. Def. Sci. J. **68**(4), 356 (2018)
11. Hasan, M., Orgun, M.A., Schwitter, R.: A survey on real-time event detection from the Twitter data stream. J. Inf. Sci. **44**(4), 443–463 (2018). https://doi.org/10.1177/0165551517698564
12. Hasan, M., Orgun, M.A., Schwitter, R.: Real-time event detection from the Twitter data stream using the TwitterNews+ Framework. Inf. Process. Manage. **56**(3), 1146–1165 (2019). https://doi.org/10.1016/j.ipm.2018.03.001
13. Imam, N.H., Vassilakis, V.G.: A survey of attacks against Twitter spam detectors in an adversarial environment. Robotics **8**(3), 50 (2019). https://doi.org/10.3390/robotics8030050
14. Khandpur, R.P., Ji, T., Jan, S., Wang, G., Lu, C.T., Ramakrishnan, N.: Crowdsourcing cybersecurity: cyber attack detection using social media. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 1049–1057 (2017)

15. Kumar, S., Liu, H., Mehta, S., Subramaniam, L.V.: From Tweets to Events: Exploring a Scalable Solution for Twitter Streams. arXiv:1405.1392 [cs] (2014)
16. Mazoyer, B., Cagé, J., Hervé, N., Hudelot, C.: A French corpus for event detection on Twitter. In: Proceedings of the 12th Language Resources and Evaluation Conference, pp. 6220–6227. European Language Resources Association, Marseille, France (2020)
17. Mazurczyk, W., Drobniak, S., Moore, S.: Towards a systematic view on cybersecurity ecology. In: Akhgar, B., Brewster, B. (eds.) Combatting Cybercrime and Cyberterrorism. ASTSA, pp. 17–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-38930-1_2
18. McMinn, A.J., Jose, J.M.: Real-time entity-based event detection for Twitter. In: Mothe, J., et al. (eds.) CLEF 2015. LNCS, vol. 9283, pp. 65–77. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24027-5_6
19. OWASP: Threat modeling (2022). https://owasp.org/www-community/Threat_Modeling
20. Petrović, S., Osborne, M., Lavrenko, V.: Streaming first story detection with application to Twitter. In: Human Language Technologies: The 2010 Annual Conference of the north American Chapter of the Association For Computational Linguistics, pp. 181–189 (2010)
21. Ritter, A., Wright, E., Casey, W., Mitchell, T.: Weakly supervised extraction of computer security events from Twitter. In: Proceedings of the 24th International Conference on World Wide Web, pp. 896–905. WWW 2015, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (2015). https://doi.org/10.1145/2736277.2741083
22. Sabottke, C., Suciu, O., Dumitras, T.: Vulnerability disclosure in the age of social media: exploiting twitter for predicting real-world exploits. In: 24th USENIX Security Symposium (USENIX Security 15), pp. 1041–1056. USENIX Association, Washington, D.C. (2015), https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/sabottke
23. Samonas, S., Coss, D.: The CIA strikes back: redefining confidentiality, integrity and availability in security. J. Inf. Syst. Sec. **10**(3), 1–25 (2014)
24. Vamvoudakis, K.G., Hespanha, J.P., Sinopoli, B., Mo, Y.: Adversarial detection as a zero-sum game. In: 2012 IEEE 51st IEEE Conference on Decision and Control (CDC), pp. 7133–7138 (2012). https://doi.org/10.1109/CDC.2012.6426383
25. Wang, X., Li, J., Kuang, X., Tan, Y.A., Li, J.: The security of machine learning in an adversarial setting: a survey. J. Parallel Distrib. Comput. **130**, 12–23 (2019). https://doi.org/10.1016/j.jpdc.2019.03.003, https://www.sciencedirect.com/science/article/pii/S0743731518309183
26. Wu, C., Li, X., Pan, W., Liu, J., Wu, L.: Zero-sum game-based optimal secure control under actuator attacks. IEEE Trans. Autom. Control **66**(8), 3773–3780 (2021). https://doi.org/10.1109/TAC.2020.3029342
27. Zhou, R., Lin, J., Liu, L., Ye, M., Wei, S.: Analysis of SDN attack and defense strategy based on zero-sum game. In: Ren, J., et al. (eds.) BICS 2019. LNCS (LNAI), vol. 11691, pp. 479–485. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-39431-8_46
28. Zhou, Y., Kantarcioglu, M., Xi, B.: A game theoretic perspective on adversarial machine learning and related cybersecurity applications. In: Game Theory and Machine Learning for Cyber Security, Chapter 13, pp. 231–269. John Wiley & Sons, Ltd (2021). https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119723950.ch13