Intrusion detection in computer systems as a pattern recognition task in adversarial environment: a critical review

Igino Corona, Giorgio Giacinto, Fabio Roli
University of Cagliari, Department of Electrical and Electronic Engineering
Piazza d'Armi, 09123 Cagliari (Italy)
{igino.corona, giacinto, roli}@diee.unica.it

The traditional Intrusion Detection Sistems (IDS) designing approach rely upon the *expert* (hand-written) definition of models describing either legitimate or attack patterns in computer systems. According to the employed intrusion detection model, an alert is produced if a pattern is not included in the model of legitimate actions patterns, or if it is included in the models of attacks. Legitimate actions (resp. attacks) may be defined as actions performed by users accessing computer system services and resources according to (resp. deviating from) the use they have been deployed for. Such actions are evaluated through their *measurable* pattern features. In addition to the need for human expertise to model legitimate or attack patterns, it is time and effort expensive to guarantee the effectiveness of these approaches (i.e., high detection rates, and low fals alarm rates). Finally, these models (especially, attack pattern models) offer poor novel (zero-day) attack detection capabilities.

To cope with these problems, the intrusion detection task has been formulated as a pattern recognition task based on machine learning algorithms. The design of a pattern recognition task can be subdivided into the following steps: (1) data acquisition, (2) data preprocessing, (3) feature extraction, (4) model selection, and (5) classification and result analysis. Let us now briefly review these steps w.r.t. the intrusion detection task. **Data acquisition** should be designed so that captured data allows distinguishing as much as possible between attacks and legitimate actions. Furthermore, it should be guranteed that statistics of feature values in training data are representative of the data analyzed during the detection (classification) phase. After acquisition, data should be **preprocessed** so that patterns that do not pertain to the target class must be deleted (noise removal), and incomplete patterns must be discarded (enhancement). The **Feature extraction** step aims at representing patterns in a feature space where the highest *discrimination* between legitimate and attack patterns is attained. Such a discrimination is then maximised through the **Model selection** step, where a machine learning algorithm automatically selects the model achieving the best discrimination between legitimate and attack patterns in the training set. Finally, the **Classification** step perform the intrusion detection task either by alerting if an observed pattern is described by an attack patterns model, usually called *signature* or *misuse*-based IDS, or by alerting if it is not described by a model of legitimate activity, usually called *anomaly*-based IDS. While the use of this paradigm allows for generalisation, i.e. detecting variants of known attacks and novel, unforeseen attacks (a.k.a. zero-day attacks), it should also be taken into account that an adversary can "pollute" the design phases in order to "mis-train" the learning system. In this paper we critically analyse the impact of an adversarial environment on these different phases of the design of an intrusion detection system, review the previous works, and point out the open issues.

Considering the *data acquisition* phase, we can identify a problem affecting network-based IDS (NIDS), i.e. IDS based on the analysis of packets passing through a network node. NIDS can be easily deployed, but they cannot exactly reconstruct events occurring on monitored host(s), due to either a lack of knowledge about host traffic processing or network topology. Aa a consequence, NIDS can either accept a packet that the destination host rejects, and reject a packet that the destination host accepts [2].In the first case the IDS analyses packets which cannot be harmful for the destination host because it rejects them, while in the second case the IDS discard packets which can be potentially harmful as they are actually processed by the destionation host. These weaknesses can be exploited by an adversary to maliciously pollute the training data, as data acquisition is usually performed by network sensors which exhibit siimilar weaknesses. The *data preprocessing* step in usually performed for designing anomaly-based IDS, by discarding known attack patterns detected in data by a misuse-based IDS. However, new attacks or other "spurious" traffic patterns which are not representative of typical traffic might still be present. To filter out such patterns, a proposed solution is the use of outlier detection techniques [1], which assume that *the great majority* of training examples are legitimate patterns. An interesting question that arises is the following: *Could attacks against the learning algorithm of an IDS be defined as "outliers", and thus removed from the training data?*. Let us consider an anomaly-based IDS that looks for buffer overflows attempts by analysing the content of FTP commands. The training phase may be accomplished by observing FTP commands content length, and training a classifier to model the normal (legitimate) FTP commands content lenght. An adversary might include FTP commands with arbitrary large content lenght, to intentionally mis-train the learner, even if these commands are not actually effective attacks from the monitored host(s) point of view. In this case, attacks that really could mis-train the system may be seen as outliers, and then discarded. On the other hand, this approach could be difficult to be performed if legitimate patterns are not the great majority of training examples, because, for example, a small training set is considered, and an attacker knows when the training phase is performed. The *feature extraction* phase is an aspect on which many works have been published in the intrusion detection literature. The aim of these works is to define a set of features that allow attaining high detection rates, and low false alarm rates. The question is whether an adversary have a somewhat control over the feature definition process. For example, the worm signature generator `Poligraph` is vulnerable to deliberate noise injection [3]. An adversary can pollute the training set made up of suspicious traffic flows to force `Poligraph` to generate signatures with wrong features.

In addition to the incorrect feature definition, the extraction of correct feature values might depend on the tool used to process the collected data. For example, [4] shows an attack against the parser of the open-source IDS Snort that avoids the correct extraction of the chunk length value related to an HTTP message chunk. HTTP version 1.1 chunked encoding allows HTTP messages to be broken up into multiple parts called HTTP chunks. Following the relative RFC, every chunk must declare the byte length of its content in the first line through `chunk-length\r\n` (value followed by Carriage Return and Line Feed chars). By adding a tab character as in `chunk-length\t\r\n` Snort v. 2.1.1 fails to acquire the `chunk-length` value, even if the HTTP server Apache v. 2.0.1 still correctly acquires such a value. Thus an attacker not only can design an evading attack, but can also influence the values of the features.

The *model selection* step has been addressed by a large number of paper, with the aim of designing the most effective algorithm to discriminate attacks from legitimate traffic. However, to the best of our knowledge, only the paper [5] addressed the case of malicious patterns in the training set in a general way. The problem is formulated in terms od an adversary that inserts malicious training examples to induce an incorrect model selection. Let us make an example of an anomaly-based IDS aimed at modeling legitimate characters in a URI of a HTTP request. The training phase can pe perfomred by considering as legitimate every character seen in a URI in the training set. An attacker might perform (non intrusive) successful requests (i.e. `/index.html?attrib='|%%35%63.+32`) using characters typically contained in intrusive requests (`/scripts/..%%35%63../winnt/system32/cmd.exe?/c+dir`) during the training phase, leading to an incorrect model selection. However, these problems can be addressed by using two or more different model(s) for each feature and correlating the corresponding outputs. It ca be easily seen that for an attacker it is more difficult to compromise simultanously different models , because it should involve the injection of suitably crafted patterns that pollute all the models. For example, if we use additioal different models for characters in a URI, it is likely that some of them identify the request `/scripts/..%%35%63../winnt/system32/cmd.exe?/c+dir` as anomalous.

Finally, the most severe problem of the *classification* phase is related to IDS over-stimulation (that is, the deliberate injection of false alarms). Considering an anomaly-based IDS, this problem is usually significant, because a pattern might be "anomalous" even if it is not related to an attack. An adversary might exploit such a weakness by flooding the IDS with patterns conceived to raise a large number of false alarms. To cope with the over-stimulation problem, alert verification has been proposed [6]. Such techniques verify whether an attack successfully succeeded or not, and thus it may be considered as a post-processing task. Let us make an example regarding an anomaly-based IDS monitoring HTTP server requests. In this case, an alert may be verified by checking if a certain HTTP request is successfully satisfied, and associated to a Web application. For example, an alert can be raised if a bad request is successfully satisfied by the server, but it does not represent a threat if no web application is associated with it.

In conclusion, we noted that the use of machine learning techniques to formulate the intrusion detection problem has been mainly focused in the definition of the "optimal" set of features, and the "optimal" learning algorithm to attain high detection rates and low false alarm rates. Until recently, very few works addressed the problem of the adversarial environment in which intrusion detection systems both learn and operate. Generally, we note that some solutions used to deal with IDS evasion or over-stimulation might be helpful to enhance the reliability of the machine learning approach. The data acquisition phase need to ensure that acquired informations are not corrupted, e.g., using a correct NIDS placement to ensure that captured data effectively reach a destination host, and a traffic normalizer to remove traffic ambiguities. The data preprocessing phase should remove outliers in training examples, while the correctness of the feature extraction procedures should be assessed, even if they are well-designed. The model selection phase needs to be further reconsidered in order to explicitly include the presence of malicious training examples. Finally, a post-processing phase perfomed as an alert verification process appears necessary to cope with inevitable classification errors.As the intrusion detection problem is clearly formulated in an adversarial environment, our opinion is that new solutions should be devised by *thinking as an adversary does*. On the other hand, current solutions are mainly countermeasures designed as a defense measure. Such new solutions could be based, for example, on defining a large number of redundant features, and different models. During training and/or detection, random subsets of features and models could be used, provided that high detection rates and low false alarm rates are produced. In such a case, for an adversary is more difficult to conceive well-crafted traffic to affect the machine learning process, simply because the "polluted" traffic should be very design-specific to be dangerous.

References

[1] Tandon, G. et al., Data Cleaning and Enriched Representations for Anomaly Detection in System Calls, Machine Learning and Data Mining for Computer Security, Springer, pp.137-156, 2006

[2] Ptacek, T. and Newsham, T., Insertion, Evasion, and Denial of Service: evading Network Intrusion Detection, Technical Report, Secure Networks Inc., 1998

[3] Perdisci, R. et al., Misleading Worm Signature Generators Using Deliberate Noise Injection, Proceedings of the 2006 IEEE Symposium on Security and Privacy, 2006

[4] Mutz, D. et al., Reverse Engineering of Network Signatures, Proceedings of the AusCERT Asia Pacific Information Technology Security Conference, Gold Coast, Australia, 2005

[5] Kearns, M. and Li, M., Learning in the presence of malicious errors, SIAM Journal on Computing, Volume 22, pp.807-837, 1988

[6] Kruegel, C., et al., Using Alert Verification to Identify Successful Intrusion Attempts, Practice in Information Processing and Communication (PIK), volume 27, pp.219-227, 2004