Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 4, April 2015, pg.768 – 774

RESEARCH ARTICLE

Security Evaluation of Pattern Classifiers in Adversarial Environments

Asharani V[1], B N Veerappa[2], Mohamed Rafi[3]

[1]P G Scholar, Department of Computer Science, UBDTCE (VTU), India

[2]Associate Professor, Department of Computer Science, UBDTCE (VTU), India

[3]Professor, Department of Computer Science, UBDTCE (VTU), India

[1] ashacssit09@gmail.com, [2] bnveerappa@yahoo.co.uk, [3] rafimohamed17@gmail.com

***Abstract*— Pattern classification systems are commonly used in adversarial applications, like biometric authentication, network intrusion detection, and spam filtering, in which data can be purposely manipulated by humans to undermine their operation. As this adversarial scenario is not taken into account by classical design methods, pattern classification systems may exhibit vulnerabilities, whose exploitation may severely affect their performance, and consequently limit their practical utility. Several works have addressed the problem of designing robust classifiers against these threats, although mainly focusing on specific applications and kinds of attacks. In this paper, we address one of the main open issues: evaluating at design phase the security of pattern classifiers, namely, the performance degradation under potential attacks they may incur during operation. We propose a framework for empirical evaluation of classifier security that formalizes and generalizes the main ideas proposed in the literature.**

Keywords: **Pattern classification, Adversarial classification, Security evaluation, Arms race, Security by design.**

I. INTRODUCTION

Pattern classification systems based on machine learning algorithms are commonly used in security-related applications like biometric authentication, network intrusion detection, and spam filtering, to discriminate between a "legitimate" and a "malicious" pattern class (e.g., legitimate and spam emails). Contrary to traditional ones, these applications have an intrinsic

adversarial nature since the input data can be purposely manipulated by an intelligent and adaptive adversary to undermine classifier operation. This often gives rise to an arms race between the adversary and the classifier designer. Well known examples of attacks against pattern classifiers are: submitting a fake biometric trait to a biometric authentication system (spoofing attack) [1]; modifying network packets belonging to intrusive traffic to evade intrusion detection systems; manipulating the content of spam emails to get them past spam filters (e.g., by misspelling common spam words to avoid their detection) [2]. Adversarial scenarios can also occur in intelligent data analysis and information retrieval; e.g., a malicious webmaster may manipulate search engine rankings to artificially promote her1 web site.

It is now acknowledged that, since pattern classification systems based on classical theory and design methods [3] do not take into account adversarial settings, they exhibit vulnerabilities to several potential attacks allowing adversaries to undermine their effectiveness [4]. A systematic and unified treatment of this issue is thus needed to allow the trusted adoption of pattern classifiers in adversarial environments, starting from the theoretical foundations up to novel design methods, extending the classical design cycle of [3]. In particular, three main open issues can be identified: (i) analyzing the vulnerabilities of classification algorithms, and the corresponding attacks [5]; (ii) developing novel methods to assess classifier security against these attacks, which is not possible using classical performance evaluation methods [4]; (iii) developing novel design methods to guarantee classifier security in adversarial environments [4].

II. RELATED WORK

A. A taxonomy of attacks against pattern classifiers

A taxonomy of potential attacks against pattern classifiers was proposed in [5] as a baseline to characterize attacks on learners. The taxonomy is based on three main features: the kind of influence of attacks on the classifier, the kind of security violation they cause, and the specificity of an attack. The attack's influence can be either **causative**, if it aims to undermine learning, or exploratory, if it targets the classification phase. Accordingly, a causative attack may manipulate both training and testing data, whereas an **exploratory** attack only affects testing data. Examples of causative attacks include work in [6], [7] while exploratory attacks can be found in [8]. The security violation can be either an **integrity** violation, if it aims to gain unauthorized access to the system (i.e., to have malicious samples be misclassified as legitimate); an **availability** violation, if the goal is to generate a high number of errors (both false-negatives and false-positives) such that normal system operation is compromised (e.g., legitimate users are denied access to their resources); or a **privacy** violation, if it allows the adversary to obtain confidential information from the classifier (e.g., in biometric recognition, this may amount to recovering a protected biometric template of a system's client). Finally, the attack specificity refers to the samples that are affected by the attack. It ranges continuously from **targeted** attacks (e.g., if the goal of the attack is to have a specific spam email misclassified as legitimate) to **indiscriminate** attacks (e.g., if the goal is to have any spam email misclassified as legitimate).

Each portion of the taxonomy specifies a different type of attack as laid out in Barreno et al. [5] and here we outline these with respect to a PDF malware detector. An example of a **causative integrity** attack is an attacker who wants to mislead the malware detector to falsely classify malicious PDFs as benign. The attacker could accomplish this goal by introducing benign PDFs with malicious features into the training set and the attack would be **targeted** if the features corresponded to a particular malware or otherwise an **indiscriminate** attack. Similarly, the attacker could cause a **causative availability** attack by injecting malware training examples that exhibited features common to benign messages; again, these would be **targeted** if the attacker wanted a particular set of benign PDFs to be misclassified. A **causative privacy** attack, however, would require both manipulation of the training and information obtained from the learned classifier. The attacker could inject malicious PDFs with features identifying a particular author and then subsequently test if other PDFs with those features were labeled as malicious; this observed behavior may leak private information about the authors of other PDFs in the training set.

In contrast to the causative attacks, **exploratory** attacks cannot manipulate the learner, but can still exploit the learning mechanism. An example of an **exploratory integrity** attack involves an attacker who crafts a malicious PDF for an existing malware detector. This attacker queries the detector with candidate PDFs to discover which attributes the detector uses to identify malware, thus, allowing her to re-design her PDF to avoid the detector. This example could be **targeted** to a single PDF exploit or

indiscriminate if a set of possible exploits are considered. An **exploratory privacy** attack against the malware detector can be conducted in the Security Evaluation of SVMs in Adversarial Environments same way as the **causative privacy** attack described above, but without first injecting PDFs into the training data. Simply by probing the malware detector with crafted PDFs, the attacker may divulge secrets from the detector. Finally, **exploratory availability** attacks are possible in some applications but are not currently considered to be of interest availability attacks are possible in some applications but are not currently considered to be of interest.

B. Arms Race: Reactive and Proactive Security(Security by Design)

Security problems are often cast as a reactive arms race, in which the system designer and the adversary attempt to achieve their goals by reacting to the changing behavior of the opponent, i.e., learning from the past. This can be modeled as the following cycle [9]. First, the adversary analyzes the existing system and manipulates data to violate its security; e.g., to evade detection, a spammer may gather some knowledge of the words used by the targeted antispam filter to block spam, and then manipulate spam emails accordingly (words like "cheap" can be misspelled as "che4p"). Second, the system designer reacts by analyzing the novel attack samples and updating the system consequently; e.g., by adding features to detect the novel attacks, and retraining the classifier on the newly-collected samples. In the previous example, this amounts to retraining the filter on the newly-collected spam, thus adding novel spam words into the filter's dictionary. This reactive arms race continues everlastingly (Fig. 1, left). However, reactive approaches do not anticipate new security vulnerabilities nor they attempt to forecast future attacks, leaving the system vulnerable to them.

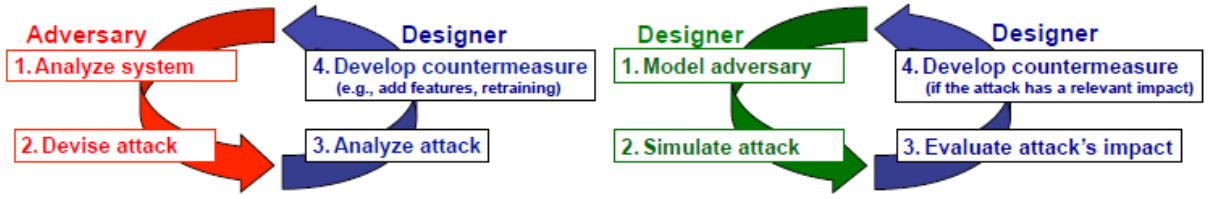

Fig 1: A schematic representation of the reactive (left) and proactive (right) arms races incurring in security applications involving pattern recognition systems.

To secure a system, a common approach used in engineering and cryptography is security by obscurity that relies on keeping secret some of the system details to the adversary. In contrast, the paradigm of security by design advocates that systems should be designed from the ground-up to be secure, without assuming that the adversary may ever find out some important system details. Accordingly, the system designer should anticipate the adversary by simulating a "proactive" arms race to (i) figure out the most relevant threats and attacks, and (ii) devise proper countermeasures, before deploying the classifier (see Fig. 1, right). This paradigm typically improves security by delaying each step of the "reactive" arms race, as it requires the adversary to spend a greater effort (time, skills, and resources) to find and exploit vulnerabilities. System security should thus be guaranteed for a longer time, with less frequent supervision or human intervention.

The goal of security evaluation is to address issue (i) above, i.e., to simulate a number of realistic attack scenarios that may be incurred during operation, and to assess the impact of the corresponding attacks on the targeted classifier to highlight the most critical vulnerabilities. This amounts to performing a what-if analysis [10], which is a common practice in security. This approach has been implicitly followed in several previous works, but never formalized within a general framework for the empirical evaluation of classifier security. Although security evaluation may also suggest specific countermeasures, the design of secure classifiers, i.e., issue (ii) above, remains a distinct open problem.

III. FRAMEWORK FOR EMPIRICAL EVALUATION OF CLASSIFIER SECURITY

We propose here a framework for the empirical evaluation of classifier security in adversarial environments. Our main goal is to provide a quantitative and general-purpose basis for the application of the what-if analysis to classifier security evaluation, based on the definition of potential attack scenarios.

A. Attack scenario and model of the adversary

The proposed model of the adversary is based on specific assumptions about her goal, knowledge of the system, and capability to modify the underlying data distribution by manipulating individual samples. It allows the classifier designer to model the attacks identified in the attack taxonomy described [5], [11]. However, in our framework, one can also incorporate application-specific constraints into the definition of the adversary's capability. Therefore, it can be exploited to derive practical guidelines for developing optimal attack strategies and to guide the design of adversarial resilient classifiers.

1) *Adversary's Goal*

According to the taxonomy presented first by Barreno et al. and extended by Huang et al. [11], the adversary's goal should be defined based on the anticipated security violation, which might be an integrity, availability, or privacy violation and also depending on the attack's specificity, which ranges from targeted to indiscriminate. Further, the adversary's goal should be defined in terms of an objective function that the adversary is willing to maximize. This allows for a formal characterization of the optimal attack strategy. For instance, in an indiscriminate integrity attack, the adversary may aim to maximize the number of spam emails that evade detection, while minimally manipulating their content [12], whereas in an indiscriminate availability attack, the adversary may aim to maximize the number of classification errors, thereby causing a general denial-of-service due to an excess of false alarms [6].

2) Adversary's Knowledge

The adversary's knowledge of the attacked system can be defined based on the main components involved in the design of a machine learning system as depicted in Fig 2. According to the five design steps depicted in Fig 2, the adversary may have various degrees of knowledge (ranging from no information to complete information) pertaining to the following five components:

(k.i) the training set (or part of it);
(k.ii) the feature representation of each sample; i.e., how real objects (emails, network packets, etc.) are mapped into the feature space;
(k.iii) the learning algorithm and its decision function; e.g., that logistic regression is used to learn a linear classifier;
(k.iv) the learned classifier's parameters; e.g., the actual learned weights of a linear classifier;
(k.v) feedback from the deployed classifier; e.g., the classification labels assigned to some of the samples by the targeted classifier.

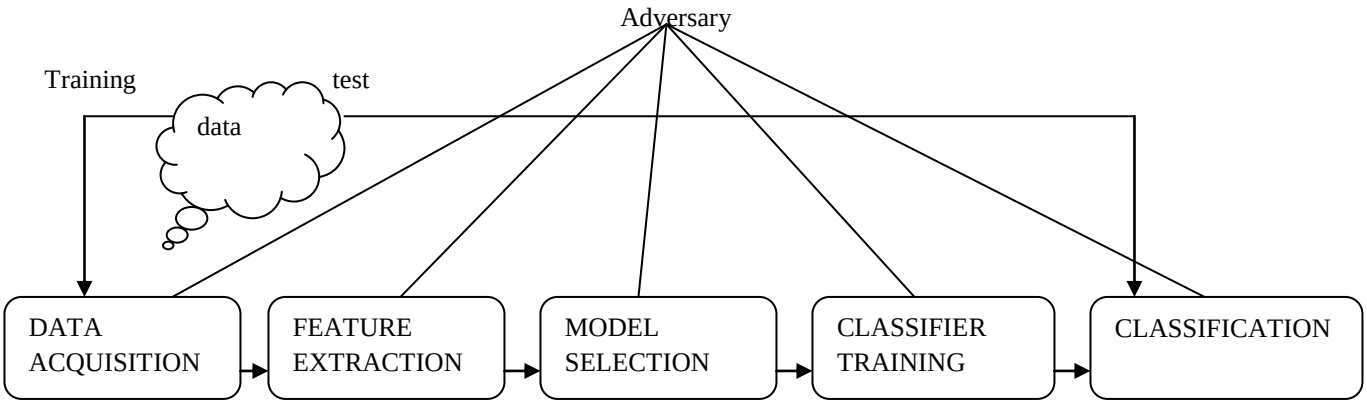

Fig. 2 A representation of the design steps of a machine learning system may provide sources of knowledge for the adversary.

These five elements represent different levels of knowledge about the system being attacked. A typical hypothesized scenario assumes that the adversary has perfect knowledge of the targeted classifier (k.iv). Although potentially too pessimistic, this worst-case setting allows one to compute a lower bound on the classifier performance when it is under attack [13]. A more realistic setting is that the adversary knows the (untrained) earning algorithm (k.iii), and she may exploit feedback from the classifier on the labels assigned to some query samples (k.v), either to directly find optimal or nearly-optimal attack instances [12], or to learn a surrogate classifier, which can then serve as a template to guide the attack against the actual classifier.

Note that one may also make more restrictive assumptions on the adversary's knowledge, such as considering partial knowledge of the feature representation (k.ii), or a complete lack of knowledge of the learning algorithm (k.iii). Investigating classifier security against these uninformed adversaries may yield a higher level of security. However, such assumptions would be contingent on security through obscurity; that is, the provided security would rely upon secrets that must be kept unknown to the adversary even though such a high level of secrecy may not be practical. Reliance on unjustified secrets can potentially lead to catastrophic unforeseen vulnerabilities. Thus, this paradigm should be regarded as being complementary to security by design, which instead advocates that systems should be designed from the ground-up to be secure and, if secrets are assumed, they must be well-justified. Accordingly, security is often investigated by assuming that the adversary knows at least the learning algorithm and the underlying feature representation.

3) Adversary's Capability

We now give some guidelines on how the attacker may be able to manipulate samples and the corresponding data distribution. As discussed in [5], [11], the adversary may control both training and test data (causative attacks), or only on test data (exploratory attacks). Further, training and test data may follow different Biggio, Corona, Nelson, Rubinstein, Maiorca, Fumera, Giacinto, Roli distributions, since they can be manipulated according to different attack strategies by the adversary. Therefore, we should specify:

(c.i) whether the adversary can manipulate training (TR) and/or testing (TS) data; i.e., the attack influence from the taxonomy in [5];
(c.ii) whether and to what extent the attack affects the class priors, for TR and TS;
(c.iii) which and how many samples can be modified in each class, for TR and TS;
(c.iv) which features of each attack sample can be modified and how can these features' values be altered; e.g., correlated feature values cannot be modified independently.

Assuming a generative model $p(X,Y) = p(Y)p(X|Y)$ (where we use p_{tr} and p_{ts} for training and test distributions, respectively), assumption (c.ii) specifies how an attack can modify the priors $p_{tr}(Y)$ and $p_{ts}(Y)$ while assumptions (c.iii) and (c.iv) specifies how it can alter the class-conditional distributions $p_{tr}(X|Y)$ and $p_{ts}(X|Y)$. To perform security evaluation according to the hypothesized attack scenario, it is thus clear that the collected data and generated attack samples should be resampled according to the above distributions to produce suitable training and test set pairs. This can be accomplished through existing resampling algorithms like cross-validation or bootstrapping, when the attack samples are independently sampled from an identical distribution (i.i.d.). Otherwise, one may consider different sampling schemes. For instance, in Biggio et al. [6] the attack samples had to be injected into the training data, and each attack sample depended on the current training data, which also included past attack samples. In this case, it was sufficient to add one attack sample at a time, until the desired number of samples was reached.

4) Attack Strategy

Once specific assumptions on the adversary's goal, knowledge, and capability are made, one can compute the optimal attack strategy corresponding to the hypothesized attack scenario; i.e., the adversary model. This amounts to solving the optimization problem defined according to the adversary's goal, under proper constraints defined in accordance with the adversary's assumed knowledge and capabilities. The attack strategy can then be used to produce the desired attack samples, which then have to be merged consistently to the rest of the data to produce suitable training and test sets for the desired security evaluation, as explained in the previous section.

IV. HOW TO USE OUR FRAMEWORK

We summarize here the steps that can be followed to correctly use our framework in specific application scenarios:

1. Hypothesize an attack scenario by identifying a proper adversary's goal and according to the taxonomy in [5][11].
2. Define the adversary's knowledge according to (k.i-v), and capabilities according to (c.i-iv).
3. Formulate the corresponding optimization problem and devise the corresponding attack strategy.
4. Resample the collected (training and test) data accordingly.
5. Evaluate classifier's security on the resampled data (including attack samples).
6. Repeat the evaluation for different levels of adversary's knowledge and/or capabilities, if necessary; or hypothesize a different attack scenario.

V. CONTRIBUTIONS, LIMITATIONS AND OPEN ISSUES

In this paper we focused on empirical security evaluation of pattern classifiers that have to be deployed in adversarial environments, and proposed how to revise the classical performance evaluation design step, which is not suitable for this purpose. Our main contribution is a framework for empirical security evaluation that formalizes and generalizes ideas from previous work, and can be applied to different classifiers, learning algorithms, and classification tasks. It is grounded on a formal model of the adversary that enables security evaluation; and can accommodate application-specific techniques for attack simulation. This is a clear advancement with respect to previous work, since without a general framework most of the proposed techniques (often tailored to a given classifier model, attack, and application) could not be directly applied to other problems.

An intrinsic limitation of our work is that security evaluation is carried out empirically, and it is thus data dependent; on the other hand, model-driven analyses require a full analytical model of the problem and of the adversary's behavior that may be very difficult to develop for real-world applications. Another intrinsic limitation is due to fact that our method is not application-specific, and, therefore, provides only high-level guidelines for simulating attacks. Indeed, detailed guidelines require one to take into account application specific constraints and adversary models.

Our future work will be devoted to develop techniques for simulating attacks for different applications. Although the design of secure classifiers is a distinct problem than security evaluation, our framework could be also exploited to this end.

REFERENCES

[1] R. N. Rodrigues, L. L. Ling, and V. Govindaraju, "Robustness *of multimodal biometric fusion methods against spoof attacks*," J. Vis. Lang. Comput., vol. 20, no. 3, pp. 169–179, 2009.
[2] D. Lowd and C. Meek, "Good *word attacks on statistical spam filters*," in 2nd Conf. on Email and Anti-Spam, CA, USA, 2005.
[3] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley-Interscience Publication, 2000.
[4] A. Kolcz and C. H. Teo, "Feature *weighting for improved classifier robustness*," in 6th Conf. on Email and Anti-Spam, CA, USA, 2009.
[5] M. Barreno, B. Nelson, A. Joseph, and J. Tygar, "*The security of machine learning*," Machine Learning, vol. 81, pp. 121–148, 2010.
[6]. Biggio, B., Nelson, B., Laskov, P.: *Poisoning attacks against support vector machines*. In: Proceedings of the 29th International Conference on Machine Learning (2012).
[7] A. A. C´ardenas, J. S. Baras, and K. Seamon, "*A framework for the evaluation of intrusion detection systems*," in Proc. IEEE Symp. On Security and Privacy. DC, USA: IEEE CS, 2006, pp. 63–77.
[8]Fogla, P., Sharif, M., Perdisci, R., Kolesnikov, O., Lee, W.: *Polymorphic blending attacks*. In: Proceedings of the 15th Conference on USENIX Security Symposium (2006).

[9]. Cauwenberghs, G., Poggio, T.: *Incremental and decremental support vector machine learning*. In: T.K. Leen, T.G. Dietterich, V. Tresp (eds.) NIPS, pp. 409–415. MIT Press (2000).
[10] S. Rizzi, "*What-if analysis*," Enc. of Database Systems, pp. 3525– 3529, 2009.
[11] Huang, L., Joseph, A.D., Nelson, B., Rubinstein, B., Tygar, J.D.: *Adversarial machine learning*. In: Proceedings of the 4th ACMWorkshop on Artificial Intelligence and Security (AISec), pp. 43–57 (2011).
[12]Lowd, D., Meek, C.: *Adversarial learning*. In: Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 641–647 (2005) .
[13]Dalvi, N., Domingos, P., Mausam, Sanghai, S., Verma, D.: *Adversarial classification*. In: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 99–108 (2004)

