Pattern Recognition Systems Under Attack: Design Issues and Research Challenges

Battista Biggio, Giorgio Fumera, and Fabio Roli

Dept. of Electrical and Electronic Engineering,
University of Cagliari,
Piazza d'Armi, 09123, Cagliari, Italy
{battista.biggio,fumera,roli}@diee.unica.it
http://pralab.diee.unica.it

Abstract

We analyze the problem of designing pattern recognition systems in adversarial settings, under an engineering viewpoint, motivated by their increasing exploitation in security-sensitive applications like spam and malware detection, despite their vulnerability to potential attacks has not yet been deeply understood. We first review previous work and report examples of how a complex system may be evaded either by leveraging on trivial vulnerabilities of its untrained components, *e.g.*, parsing errors in the pre-processing steps, or by exploiting more subtle vulnerabilities of learning algorithms. We then discuss the need of exploiting both reactive and proactive security paradigms complementarily to improve the security by design. Our ultimate goal is to provide some useful guidelines for improving the security of pattern recognition in adversarial settings, and to suggest related open issues to foster research in this area.

1 Introduction

Pattern recognition systems have been increasingly exploited in security-sensitive applications like spam, malware, and intrusion detection, motivated by the growing number, sophistication and variability of recent attacks and security threats [70, 31, 51, 47, 35, 63, 61]. However, these techniques have not been originally designed to cope with intelligent, adaptive attackers that may purposely modify the input data to mislead the outcome of an automatic analysis, thus breaking the underlying assumption of *data stationarity* (*i.e.*, that training and test data follow the same distribution) [65, 20, 41, 6, 24]. This has revealed a number of intrinsic vulnerabilities that can be exploited to cause different security violations, including denial of service and missed detection of malicious samples or events. In contrast, countermeasures and novel algorithms have been also proposed to improve system security against well-crafted

attacks [31, 24, 64, 28]. This will not stop adversaries from attempting to find novel ways of misleading such defense systems, triggering a phenomenon commonly modeled as an arms race.

In this work, we revise under an engineering viewpoint previous work on the design of pattern recognition systems in adversarial settings [65, 20, 41, 6, 24]. Most of the proposed defense strategies follow the paradigm of *proactive security*: they attempt to prevent potential attacks, or mitigate their impact, before they occur, by incorporating knowledge of the attacker's strategy in the designed algorithm or defense mechanism. In real-world applications, however, deployed systems still follow a *reactive* approach: they are only updated when novel attacks are observed, by adding novel features (if required), and, more often, by retraining the system using newly-collected data. We argue that both paradigms should be exploited complementarily, for thoroughly assessing security against carefully-targeted attacks, and of improving it security by design. We also underline that the security of a complex pattern recognition system crucially depends on that of its *weakest link* [3, 43], *i.e.*, its most vulnerable point (a well-known concept in computer security), which may not necessarily be a complex, trained component (*e.g.*, a classifier). We report examples of how complex systems may be evaded by leveraging on trivial vulnerabilities of untrained components, *e.g.*, parsing errors in the pre-processing steps, but also examples of how more subtle vulnerabilities of learning algorithms can be exploited to the same end. Our main goal is to provide useful guidelines for improving the security of pattern recognition in adversarial settings, and to suggest related open issues to foster research in this area.

In Sect. 2, we define reactive and proactive arms races, using spam filtering as an example. In Sect. 3, we describe the architecture of a pattern recognition system that will be exploited later to discuss potential vulnerabilities. A model of the adversary, that can be exploited to identify them, and for proactively evaluating security, is presented in Sect. 4, with some examples from our recent work. Proactive defenses derived by our adversary's model, and reactive defenses are discussed in Sect. 5. Conclusions and future applications are finally discussed in Sect. 6.

2 Pattern Recognition for Computer Security

We introduce the problem of designing and deploying pattern recognition systems in security applications through an example related to the spam filtering task. This leads us to define the concepts of reactive and proactive security.

Motivation and Trends. Since the 90s, the variability and sophistication of computer viruses and attack threats increased, in response to the growing complexity and amount of vulnerable attack points of security systems. Since automatic tools for designing novel variants of attacks can be easily obtained and exploited by not very skilled attackers, and a flourishing underground economy strongly motivates them, an exponential proliferation of malware and other threats has been recently observed. To cope with such malicious data, including

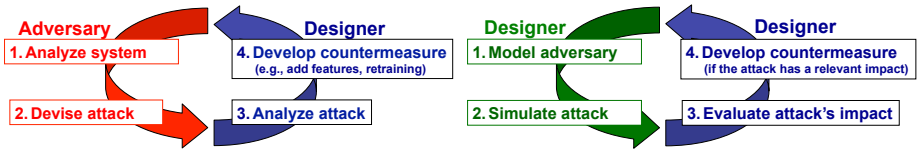

Figure 1: A schematic representation of the reactive (left) and proactive (right) arms races incurring in security applications involving pattern recognition systems.

never-before-seen attacks, machine learning has been increasingly adopted to complement the earlier rule-based systems (*e.g.*, signature-based systems based on string matching): the latter filters known attacks, while the former can detect their variants, and novel attacks. However, machine learning turned out to be not a panacea.

A known example of that is spam filtering. Spam e-mails originally conveyed their message as "clean" body text. Rule-based spam filters were firstly adopted, to detect the presence of "spammy" words. Machine learning (*i.e.*, text classification) was later introduced to perform a more sophisticated analysis of text messages. Spammers thus started "obfuscating" their text, *e.g.*, by misspelling spammy words and adding random vocabulary words, to avoid detection by simple rules and text classifiers, respectively [70, 47, 51]. During 2005, spammers devised *image-based spam* (or image spam) [13, 4], *i.e.*, rendering the spam message into attached images to evade the textual-based analysis. Due to the massive volume of image spam sent in 2006 and 2007, researchers and companies developed countermeasures, based on signatures of known spam images, and on extracting text from suspect images by OCR tools, for standard spam detection. Spammers reacted by *randomly* obfuscating images with *adversarial* noise, both to evade signature-based detection, and to make OCR-based detection ineffective. Researchers responded with approaches based on machine learning techniques using visual features, to discriminate between images attached to spam and to legitimate e-mails. Image spam volumes have since declined, but spammers kept introducing novel tricks.

Reactive and proactive security. As highlighted by the spam filtering story, 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 [20]. 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 anti-spam 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 spammy words into the

filter's dictionary (*e.g.*, "che4p"). 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. Computer security guidelines accordingly advocate a *proactive* approach in which the designer should also attempt to *anticipate* the adversary's strategy by (i) identifying the most relevant threats, (ii) devising proper countermeasures, when required, and (iii) repeating this process *before* system deployment. To this aim, one can simulate attacks based on a model of the adversary, to complement the reactive arms race (Fig. 1, right). While this does not account for unknown or changing aspects of the adversary, it can improve security by delaying each step of the *reactive* arms race, as it should force the adversary to exert greater effort (time, skills, and resources) to find new vulnerabilities. Accordingly, the resulting systems should remain effective for a longer time, with less frequent supervision or human intervention and with less severe vulnerabilities. However, this approach requires a thorough and systematic revision and re-engineering of traditional design methodologies, since state-of-the-art pattern recognition and machine learning techniques do not take into account the presence of malicious adversaries, and rely on the assumption that training and testing data follow the same distribution (or that distribution drift, if any, is not maliciously driven). Although proactive security has been implicitly pursued in most of previous work, it has only recently been formalized within a general framework for the empirical evaluation of pattern classifier's security[20], which we summarize in the next section.

3 Pattern Recognition Systems: Outside and Inside Worlds

Fig. 2 shows the architecture of pattern recognition systems, with reference to classification tasks. We will use it to discuss the vulnerabilities that an adversary could exploit, and the possible countermeasures. To this aim, here we distinguish between the "outside" world (everything that is out of designer's control, including the physical process that generates data, and the adversary) and the "inside" world (the system's components that are under the control of the designer).

The Outside World. Input samples (*e.g.*, images) are generated by a physical system, modeled by a (unknown) distribution $p(X, Y)$, where the random variables X and Y denote respectively a sample (*not* its representation, *e.g.*, in terms of a feature vector) and its class label. The classical underlying assumption is that of stationarity, *i.e.*, the samples collected for system design (from now on, *training* samples) and the ones processed during operation (*testing* samples) come from the same $p(X, Y)$. Non-stationarity has also been considered, modeled as $p(X, Y; t)$, t being a parameter denoting time, but only of non-adversarial nature. Many security applications involve such a kind of non-stationarity: the topics of legitimate e-mails evolve over time, the con-

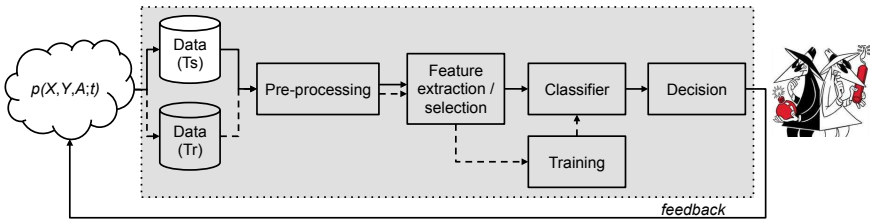

Figure 2: Architecture of a typical pattern recognition system deployed in a real-world application, where training (Tr) and test (Ts) data are drawn from a changing distribution $p(X, Y, A; t)$.

tent of network HTTP traffic changes depending on the Web sites visited by the users, biometric traits like faces are subject to aging, *etc.* However, the presence of intelligent, adaptive adversaries who exploit their knowledge of the targeted system to evolve their attack strategy over time, introduces a further source of *adversarial* non-stationarity, that can be modeled by introducing a random variable A, as $p(X, Y, A; t)$[41, 20] (see Fig. 2).

The Inside World. The system components under the designer's control are the following [33]. *Data collection* for system design is typically done off-line before deployment. It can also occur during operation, when online learning techniques are used. It can include verification of ground truth labels, and data cleaning, using automatic or semi-automatic techniques involving human supervision. For instance, in malware detection systems malware samples collected through "honeypots" are clustered (before pre-processing and feature extraction) for reducing the training set size, and for unsupervised ground truth assessment before human supervision[63, 2, 61]. *Pre-processing* of training or testing samples is then carried out; *e.g.*, images of biometric traits can be normalized (change of resolution, cropping, rotation, *etc.*); e-mails and network packets are parsed to extract structural components like e-mails' header, body and attachments, and packet header, payload and footer. *Feature extraction* is carried out on parsed samples, and may be followed by a *feature selection* step, either with or without supervision. A *learning algorithm* is then applied at design stage to a labelled training set (comprising all or a subset of collected data) for building a classifier. When online learning is used, the learning algorithm is periodically rerun on novel data during operation. At operation phase, unlabeled test samples are fed to the *classifier*, which usually produces a real-valued score which is then fed to a decision stage. Eventually, a *decision rule* assigns a label (usually, Malicious or Legitimate) to an input sample, based on a thresholding strategy on the classifier's score, whose parameters are set during design.

During system design, the above steps are carried out sequentially. Several cycles can occur, if the estimated system performance is not satisfactory, restarting from any of the above steps [33]. Further, some components may be deployed in different physical locations, and thus their communication channels may be affected by the adversary; *e.g.*, in systems requiring remote authentication, the

pre-processing and feature extraction steps can be implemented remotely on a client's personal computer. This introduces different vulnerabilities (not directly related to the above components), increasing the *attack surface*, *i.e.*, the set of vulnerable points.

In the above context, a useful distinction between two kinds of system failures (*e.g.*, misclassifications and denial of service) can be made, generalizing the one proposed for biometric systems [42]. *Intrinsic failures* (aka "zero-effort" failures) are statistical errors not caused by a deliberate action made by the adversary. *Failures due to adversarial attacks* are due instead to deliberate efforts aimed at misleading the system. The latter ones can be caused by exploiting vulnerabilities that all the above components potentially exhibit, besides the ones already known. For instance, note that all components depend on the analysis of data collected for system design (or even during operation), which may be manipulated by an adversary. In the next sections we will discuss how to systematically analyze the vulnerabilities of pattern recognition systems, pointing out some potential ones in Sect. 4.1, and how to re-engineer their design cycle, using a proactive approach.

4 Wearing the Black Hat: Understanding Attacks

Here we describe a model of the adversary adapted from our recent work [20, 10, 65]. It builds on a well-known attack taxonomy [6, 41], and can be exploited proactively for evaluating vulnerabilities and improving security by design (see Fig. 1, right).

The attack taxonomy proposed in [6, 41] categorizes attacks against learning algorithms along three axes: (*i*) the *attack influence*, which can be **exploratory**, if the adversary can only manipulate the test data, or **causative**, if both training and test data can be modified; (*ii*) the *attack specificity*, which ranges from **targeted** to **indiscriminate**, depending on whether the attack is focused on the classification of a specific subset of samples or of any of them; and (*iii*) the *security violation* caused by the attack, which can be an **integrity** violation, if it allows the adversary to access a restricted service or resource protected by the system (*e.g.*, an impostor may gain access to a genuine client's account [42, 64, 20], or a spam email may reach a protected user's inbox [70, 51, 47, 17]), an **availability** violation, if it denies legitimate users' access or compromise normal system operation (*e.g.*, causing legitimate emails or network traffic to be misclassified as spam or intrusive [55, 66]), or a **privacy** violation, if it allows the adversary to obtain confidential information about the classifier or its protected resources (*e.g.*, the clients' templates in a biometric recognition system [1, 36, 54]).

Our model incorporates these aspects to give clearer guidelines on how to hypothesize the attacker's goal, knowledge of the targeted system, and her capabilities of manipulating data or of compromising some of the system's components.

Goal. It is defined according to the desired *security violation* and *attack specificity*, *i.e.*, axes (*iii*) and (*ii*) of the attack taxonomy in [6, 41].

Knowledge of each system component can be *perfect* or *limited*. It includes the kind of decision function (*e.g.*, a linear SVM), the classification function and the trained parameters (*e.g.*, the feature weights of a linear classifier), and the available feedback, if any (*e.g.*, the class labels assigned to some test samples). Feedback on the classifier's decisions can be exploited to improve the attacker's level of knowledge [50, 58, 57, 10, 60]. The attacker may however not only know *how* these components are implemented (as discussed in our previous work), but also *where* they are deployed, and *when* they operate. For example, knowing how the training data is being collected, from which locations/servers and at what time, may allow her to *poison* them by injecting carefully-crafted samples [22, 46, 59], and to collect similar data to train a good approximation of the targeted classifier system, which can be used in turn to evade detection [10]. She may also know whether and how *communication channels*, if any, are used (*e.g.*, how a remote biometric authentication module connects to the main server [62, 42]); and what *security protocols* are used by human supervisors and operators to interact with the automatic classification system [42] (*e.g.*, usually supervisors of intrusion detection systems manually inspect log files to monitor attack detection and false alarms, and validate decisions).

Capability. It refers to how the adversary can affect training and test data. At an abstract level, it is defined as the *attack influence*, *i.e.*, axis (*i*) of the attack taxonomy in [6, 41]. It includes assumptions on which features can be manipulated, and to what extent, taking into account application-specific constraints (*e.g.*, some features may be correlated, some may compromise the functionality of malicious samples, if modified, *etc.* [31, 47, 35, 64]). To get a more complete view of the attack surface, Fig. 2 suggests that, besides *data manipulation*, the attacker may directly compromise each component of the system, including communication channels and each module's internals. For instance, if pre-processing and feature extraction are performed remotely, and the results are sent through a non-secure channel, she may compromise the latter and perform a man-in-the-middle attack to cause a number of misclassifications, by manipulating the output of the feature extractor instead of creating samples that exhibit certain feature values. She may even override the classifier's decision, if the whole system is compromised, *e.g.*, by a Trojan horse [62, 42]. She may also exploit weaknesses in human-dependent processes or security protocols, by leveraging on *social engineering* techniques, or on coercion or collusion with a privileged user or an administrator [62, 42].

Attack strategy. Once the adversary's goal, knowledge and capability are defined, an optimal attack strategy should be defined. If the attack only involves data manipulation, it can be formalized as an optimization problem that aims to meet the goal within a bounded or minimal effort, *e.g.*, evading detection while minimizing the number of modifications to some non-obfuscated attack samples [20, 10, 31, 50], or performing a computer intrusion while taking care of covering its traces. Instead, if the attacker can directly compromise some of the system components, or the security protocols, then the attack can be

trivially executed. Note that the *weakest link* of pattern recognition systems is not necessarily the learning or classification component; *e.g.*, they may be straightforwardly evaded by leveraging on vulnerabilities in the pre-processing (*i.e.*, file parsing) steps [43, 52].

A thorough proactive security evaluation, considering any kind of possible attack, may be unfeasible, and designing the corresponding countermeasures may be even more difficult [7, 11]. A good system designer should thus only develop proactive defenses against attacks that can be considered more relevant or threatening in the near future, which we believe our framework helps to identify.

4.1 Previous Work on Adversarial Attacks

Most previous work investigating attacks make some assumptions on the adversary's goal, knowledge and capabilities, either implicitly or explicitly, and can be recast in our framework as done for the attack taxonomy in [6, 41]. Other works hypothesize a simpler model of the attacker, which can be included in ours as well (see work cited in the previous section). The following examples will help understanding how our model can be exploited to identify and analyze non-trivial attacks.

4.1.1 Poisoning Adaptive Biometric Face Verification

The application of our framework led us to highlight a novel vulnerability of adaptive face recognition systems [21, 12]. They deal with temporal variations of the clients' faces by exploiting images acquired during system operation. Template self-update is the simplest approach, inspired by semi-supervised learning techniques: a user's template gallery is periodically updated using samples assigned with high confidence to the same identity. An *attacker* may however exploit it to compromise the stored templates by a *poisoning* attack, *i.e.*, submitting a suitable sequence of fake (or *spoofed*) faces to the camera (*e.g.*, obtained by printing a face image on paper [8]) while claiming the identity of a *victim* user. This may eventually replace some of the victim's templates with other desired face images, that may be sufficiently different from the former, to deny access to him; or similar to the attacker's images, allowing her to impersonate the victim without using any fake trait. We now give an example of the optimal poisoning attacks we derived in [21, 12], that *minimize* the size of the fake face sequence under perfect or limited knowledge.

Users are authenticated by computing a similarity score $s(F, T)$ between the submitted face image F and the stored template T of the claimed identity: if $s(F, T) > t_{\mathrm{a}}$ (a pre-defined acceptance threshold), the claimed identity is authenticated as genuine, otherwise it is rejected as an impostor attempt. In our example, the *unique* template T of each client is obtained by averaging $n = 5$ distinct face images acquired during enrollment, and it is self-updated during operation as a moving average using face images F such that $s(F, T) > t_{\mathrm{u}}$, being t_{u} a pre-defined update threshold (typically, $t_{\mathrm{u}} > t_{\mathrm{a}}$).[1] We assume that the

[1] Despite self-update of an averaged template has been proposed for face verification (see,

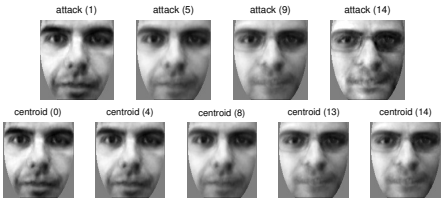

Figure 3: Attack samples (top) and victim's template (bottom) for poisoning with *limited* knowledge, at different iterations.

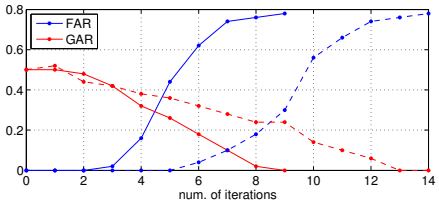

Figure 4: FAR and GAR for poisoning with *perfect* (solid lines) and *limited* (dashed lines) knowledge, at different iterations.

adversary's **goal** is to impersonate the victim without using fake traits, by replacing his template while minimizing the number of submitted fakes (*queries*). We consider both perfect and limited **knowledge**; in the former case, the attacker knows the victim's templates, the feature representation, the verification and update algorithm, and the corresponding thresholds; in the latter, more realistic case, she does not know the template, but is able to get a similar image (*e.g.*, from social networks) that meets the update condition, successfully starting the poisoning attack. **Capability:** she can submit a number of fake faces to get access to the victim's template gallery, *i.e.*, to a portion of the *training* data.

We refer the reader to [21, 12] for the computation of the optimal attack. For a specific attacker-victim pair, Fig. 3 shows how the victim's template is updated by a limited knowledge attack. Fig. 4 shows the corresponding behavior of the False Acceptance Rate (FAR, the probability of the attacker accessing the system impersonating the victim) and the Genuine Acceptance Rate (GAR, the probability of the victim correctly accessing the system). In case of perfect knowledge less queries are required, coherently with theoretical bounds [46]. In any case, the victim's account can be violated with high probability even when the template is partially compromised, as shown by the significantly high FAR value after half of the queries. As a side-effect the GAR quickly decreases, *i.e.*, the victim is denied access.

A possible countermeasure against this attack, which is part of our ongoing work, is to model the changes induced by legitimate template updates to detect *anomalous* updates, like the sudden, biased drift induced by a poisoning attack. This *proactive defense* is an example of *data sanitization* [28], as discussed in Sect. 5.1.

4.1.2 Poisoning HTTP-based Malware Clustering

In [23] we investigated the poisoning of a simplified version of an HTTP-based behavioral malware clustering system [61]. It aims at obtaining a *small* set

e.g., [21] and references therein), multiple templates are often used for each user. It is nevertheless possible to poison these systems with a similar attack to that exemplified in this work, as discussed in [12].

of *compact* malware clusters to aid the automatic generation of network signatures that can detect botnet command-and-control (C&C) and other malware-generated communications at the network perimeter. Each malware is mapped as a six-dimensional feature vector: number of GET and POST requests, average URLs length, average number of parameters in the request, average amount of data sent by POST requests, average response length. We developed a poisoning attack against the single-linkage hierarchical clustering, which is used to identify malware families.

Goal: causing a denial of service. Being clustering unsupervised, the effectiveness of poisoning can not be assessed with respect to any ground-truth labels. We then reformulated the goal as that of maximizing a given distance function between the clusterings obtained in the presence and in the absence of poisoning. We assumed perfect **knowledge**: the input data, their feature values, and the internals of the clustering algorithm are known to the attacker. **Capability:** since *poisoning* samples do not need to preserve any malicious functionality, the attacker can add any number of them to the initial dataset, with the only constraint on their feature values given by feature normalization. The optimal **attack strategy** can be therefore formulated as $\max_{\mathcal{A}} d_{\mathrm{c}}(\mathcal{C}, f_{\mathcal{D}}(\mathcal{D} \cup \mathcal{A}))$, being $\mathcal{C} = f(\mathcal{D})$ the clustering output on the initial, untainted dataset $\mathcal{D}$, $\mathcal{A}$ the set of poisoning attacks, $f_{\mathcal{D}}(\mathcal{D} \cup \mathcal{A})$ the clustering output on the points in $\mathcal{D}$ when the tainted data $\mathcal{D} \cup \mathcal{A}$ is clustered, and d_c a distance measure between clusterings. We refer the reader to [23] for details on the (approximate) solution of this optimization problem. We considered three different greedy heuristics tailored to single-linkage hierarchical clustering, named *Bridge (Best)*, *Bridge (Hard)*, and *Bridge (Soft)*: poisoning samples are added one at a time, until $|\mathcal{A}|$ points are added. Their underlying idea was to consider a number of candidate attack points that attempt to *bridge* each pair of adjacent clusters. *Bridge (Best)* evaluates the objective function for each candidate by re-running the clustering algorithm with the candidate attack point, and chooses the best attack. *Bridge (Hard)* and *Bridge (Soft)* approximate the clustering result without re-running the clustering algorithm for each candidate attack point, to reduce the computational complexity. They respectively approximate the clustering output using deterministic and probabilistic sample-to-cluster assignments, assuming that each candidate will effectively merge the two corresponding clusters.

In the experiments we used 1,000 malware samples collected from different malware sources and commercial malware feeds [23, 61]. The attacker gradually adds poisoning samples to the untainted data, up to 5% of the data. We compared the proposed heuristics with two greedy random attack strategies, named *Random* and *Random (Best)*. The former chooses a random attack point at each iteration, while *Random (Best)* chooses the best attack point among a set of randomly-selected candidates, by re-running the clustering algorithm for each candidate. For a fair comparison, the number of candidates evaluated by *Random (Best)* equals that of the candidate points evaluated by *Bridge (Best)*. The results, summarized in Fig. 5, show that the effect of the attack is to increase quickly the attacker's objective, which amounts to fragmenting the initial clustering into smaller, *heterogeneous* clusters (*i.e.*, clusters made of points ini-

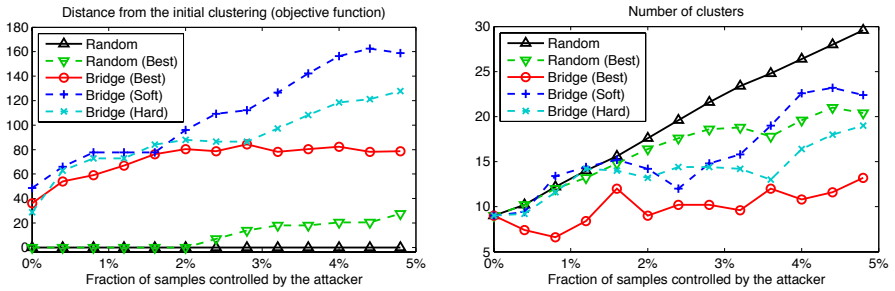

Figure 5: Results for the poisoning attack on malware clustering, for different attack heuristics: variation of the objective function $d_c(\mathcal{C}, f_{\mathcal{D}}(\mathcal{D} \cup \mathcal{A}'))$ (left); number of selected clusters as the fraction of samples controlled by the adversary increases (right).

tially belonging to different clusters). Our attack strategies are superior to the random-based ones, which witnesses the soundness of our heuristic solutions. These results underline that single-linkage hierarchical clustering may be itself a vulnerable point (and, perhaps, the weakest link) in a more complex pattern recognition system. This potential weakness not only demands for a more systematic security assessment, but, potentially, also for clustering algorithms with improved security properties.

5 Wearing the White Hat: Implementing Countermeasures

In this section we leverage on our adversary's model to discuss how attacks similar to those exemplified in the previous section, along with less complicated ones, can be countered, highlighting some guidelines for improving the design of pattern recognition systems in adversarial settings. In particular, we advocate that both proactive and reactive security paradigms should be exploited complementarily.

5.1 Proactive defenses

The main proactive defense strategies proposed thus far for pattern recognition systems come from the field of *adversarial machine learning*. They can be categorized according to two well-known paradigms in security, *i.e.*, *security by design* and *security by obscurity*, as discussed below.

Secure Learning and Security By Design. According to the paradigm of security by design, a system should be designed from the ground up to be secure. Based on this idea, modifications to learning algorithms and novel learning techniques that explicitly take into account a specific kind of adversarial data manipulation have been proposed in several recent work.

Game-theoretical approaches achieve this goal by modeling the interactions between the attacker and the classifier as a *game* in which players aim to max-

imize their own payoff function. The adversary's goal is to evade detection by minimally manipulating some attack samples, while the classifier is retrained to correctly classify them. Since the attack samples have to preserve some malicious or intrusive functionality (like the advertisements conveyed by spam emails, and valid exploits carried by malware), they can not be modified at will, and this can be formally taken into account by defining specific constraints into the definition of the attacker's strategy. The aforementioned procedure is iterated until a game equilibrium point is reached, *i.e.*, when no player can improve his payoff function unilaterally by playing a different strategy. This approach has been initially investigated in the seminal work of Dalvi *et al.* [31]; then, more rigorous approaches have introduced Nash and Stackelberg games for secure learning, deriving formal conditions for the corresponding equilibrium strategies [25, 45, 49, 26, 24]. Although these approaches seem quite promising, understanding whether and to what extent the resulting attacker's strategies are well-representative of realistic, practical scenarios remains an open issue [71, 30]. The main reason is that adversarial classification is not a game with well-defined rules such as board games (*e.g.*, chess), and, thus, the *real* attacker's objective may not even correspond to a proper payoff function. It may therefore be interesting to verify, in a *reactive* manner, whether real attackers behave as hypothesized, and exploit feedback from observed manipulations on real attacks to improve the definition of the attacker's strategy. Another issue is that most formulations (*e.g.*, Nash and Stackelberg games) do not account for limited knowledge of the attacker's objective function (that may reflect in turn the limited attacker's knowledge about the classifier) explicitly. When they do (see, *e.g.*, Bayesian games [38]), uncertainty on the payoff function is simply modeled with a prior distribution over its parameters (and, thus, the payoff function is still essentially known). Another relevant problem is the scalability of their training procedure to large datasets.

Learning with invariances is essentially a *minimax* approach in which the learning algorithm is modified to minimize the maximum loss incurred under worst-case manipulations of the attack samples. In particular, different variants of the SVM learning algorithm have been proposed to account for a maximum number of worst-case feature manipulations, including deletion, insertion, and rescaling [37, 68, 32].

Other approaches make different assumptions on the attacker's strategy and exploit different models to build secure classification techniques. A version of secure SVMs and Relevance Vector Machines (RVMs) has been proposed in [74, 73]. Similarly to the problem of learning with invariances, these classifiers aim to minimize the worst-case loss under a given attacker's model. Torkamani and Lowd [69] have faced the problem of collective classification in adversarial settings, where the goal is to jointly label a set of interconnected samples (*e.g.*, cross-linked web pages). All the aforementioned works allow us to learn secure *discriminative* classifiers by assuming that the attack samples are modified according to a given attack strategy. To learn secure *generative* classifiers, instead, the *distribution* of the manipulated attack samples should be explicitly modeled, as proposed in [19, 64]. This avoids one to define a specific strategy

for manipulating the attack samples. These approaches have been applied especially to counter *spoofing attacks* to biometric systems, *i.e.*, impostor attempts to impersonate genuine users using counterfeited traits (*e.g.*, gummy fingers). Notably, all these techniques achieve security against *evasion* attempts by assuming that only the malicious class can be manipulated in certain ways and to some extent: this results in decision functions that tend to "enclose" the legitimate class more tightly, which in turn requires to trade-off between the security against potential attacks and the number of misclassified legitimate samples.

While most works focused on countering evasion attacks at test time (which do not involve manipulating the training data), some work dealt with the *security of the training process* in the presence of training data contamination, which may occur when the system is updated with data collected during operation [56, 55, 28, 66]. To significantly compromise the training phase of a learning algorithm, an attack has to be exhibit some characteristics that are different from those shown by the rest of the training data, otherwise it would have no impact at all. Therefore, most of the training attacks can be regarded as *outliers*, and countered either by data sanitization (*i.e.*, outlier detection) [28] or by exploiting *robust* statistics [40, 53] to mitigate the outliers' impact on learning (*e.g.*, robust principal component analysis [66, 29]). Notably, in [27] the robustness of SVMs to training data contamination has been formally analyzed under the framework of Robust Statistics [40, 53], highlighting that bounded kernels and bounded loss functions may significantly limit the outliers' impact on classifier training.

Another relevant aspect for pattern recognition systems in adversarial settings is that they should be *privacy preserving*. Although some attacks can violate a system's privacy (see, *e.g.*, hill-climbing attacks to recover the clients' templates in biometric recognition systems [1, 36, 54]), their practical feasibility is still debatable. Nevertheless, some privacy-preserving classifiers have been already proposed; in particular, differentially-private SVMs [67], that implement the privacy-preserving mechanism of Dwork [34] to protect information about the training data by randomizing the SVM's decision function.

Multiple Classifier Systems (MCSs) have been exploited for different purposes in adversarial settings, in particular, to improve security against evasion attempts [47, 17, 15, 16, 18] and against poisoning of the training data [9]. MCSs indeed provide an easy way to obtain complex non-linear classification functions which are difficult to reverse-engineer while being easy to update and maintain.

The following *open issues* can be finally highlighted. (i) Adversarial data manipulation can also be seen as a particular kind of noise. This suggest to investigate the connections between secure learning techniques and *classifier regularization*, by formally characterizing adversarial noise, following the intuition in [72] related to SVMs and noise-based classifier regularization (see also [56] for an example). Secure learning models with high computational complexity (*e.g.*, game-theoretical models) may be thus revisited to identify suitable, lightweight approximations based on regularizing the objective function in an ad hoc manner. (ii) The security properties of the learning algorithm and the classifier's decision function should be considered independently from those ex-

hibited by the chosen feature representation. Security of features should be considered as an additional, important requirement: features should not only be discriminant, but also *robust* to manipulation, to avoid straightforward classifier evasion, *e.g.*, by mimicking feature values exhibited by legitimate samples. Although this is mainly an application-specific issue, since features are usually designed by domain experts, there is room for a theoretical investigation of whether and how feature correlation and redundancy may make evasion more difficult, and what problems may arise from learning in high-dimensional feature spaces. Intuitively, in the latter case, the adversary has more degrees of freedom to manipulate her samples, thus classifier evasion and poisoning may be more effective.

Information Hiding, Randomization, and Security by Obscurity. These proactive defenses, also referred to as *disinformation* techniques in [6, 5, 41], follow the paradigm of *security by obscurity*, *i.e.*, they rely on hiding information to the attacker to improve system security. Some specific examples have been suggested by R. Lippmann's talk at the 2013 Dagstuhl Perspectives Workshop on Machine Learning Methods for Computer Security [44], including: (i) randomizing collection of training data (collect at different timings, and locations); (ii) using difficult to reverse-engineer classifiers (*e.g.*, MCSs); (iii) denying access to the actual classifier or training data; and (iv) randomizing the classifier's output to give imperfect feedback to the attacker. We have investigated a specific implementation of the latter approach in [14]; however, it is still an open issue to understand whether randomization may be effective also against attacks that can be exploited to reverse-engineer the classification function [50, 57, 10], and whether and how it can implement privacy-preserving classification [67].

5.2 Reactive defenses

System security can also be improved *reactively*, by *learning from the past*, which can sometimes be more effective and convenient than a pure proactive strategy aimed at mitigating the risk of *future* attacks [7, 11]. This requires one to: (i) timely detect novel attacks, (ii) frequently retrain the classifier, and (iii) verify whether classifier decisions are consistent with the training data and the corresponding ground-truth labels. In practice, collaborative approaches and "traps" are used to identify novel security threats. For instance, collaborative spam filtering allow end-users to share signatures of newly-spread spam, and *honeypots* are used to "trap" novel malware samples and analyze them in controlled settings. The classifier should then be updated, typically by retraining it on the newly-collected data, and adding specific features, if required. While this procedure is currently carried out manually, we argue that it should be automated to some extent to act more promptly when required; *e.g.*, automatic *drift* detection techniques [48] may be exploited to detect changes in the data evidence $p(X)$. The correctness of classifier decisions should (partly) be verified by expert domains as well. This opens the issue of how to consider a more involved and coordinated presence of "humans in the loop" to supervise pattern

recognition systems, and, if needed, to fix some of its (potentially corrupted) functionalities.

6 Conclusions and Future Applications

In this paper we have presented an overview of work related to the security of pattern recognition systems with the goal of providing useful guidelines on how to improve their design and assess their security against well-crafted, specific attacks. We believe that our paper provides a useful introduction, both for system designers and researchers, on the main issues and research challenges related to the design of pattern recognition systems for security-sensitive tasks. Further, this work, and, in particular, the proposed attacker's model can be a convenient resource also to identify and prevent potential, very sophisticated future attacks. By reasoning on the (vulnerable) components of a pattern recognition system, for instance, one may expect future attacks to target the feature selection phase through an ad hoc training data manipulation. This trend should not be underestimated, as also witnessed by the past arms races: recent attacks, as the two exemplified in this paper, may be very sophisticated and effective. This may become a very relevant threat in the next years, since nothing prevent attackers to exploit machine learning in an *offensive* way, *i.e.*, as a tool for evading pattern recognition systems [10].

To conclude, we would like to point out that other emerging applications, not necessarily related to security tasks, may also exhibit an adversarial behavior, therefore requiring a revisited design of pattern recognition systems as that advocated in this paper; for instance, machine learning is increasingly being used for user authentication, in computer vision and forensics [44], for sentiment analysis and market segmentation [39]. So far, however, the analysis of carefully-crafted attacks and the security of the corresponding techniques have not yet been systematically investigated in these research areas. This is a further motivation behind continuing to push research on the security of pattern recognition systems.

Acknowledgments. This work has been partly supported by the project "Security of pattern recognition systems in future internet" (CRP-18293) funded by Regione Autonoma della Sardegna.

References

[1] A. Adler. Vulnerabilities in biometric encryption systems. In T. Kanade, A. K. Jain, and N. K. Ratha, eds., *5th Int'l Conf. on Audio- and Video-Based Biometric Person Authentication*, vol. 3546 of *LNCS*, pp. 1100–1109, 2005. Springer.

[2] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon. From throw-away traffic to bots: Detecting the

rise of DGA-based malware. In *21st USENIX Security Symp.*, pp. 491–506, 2012. USENIX.

[3] I. Arce. The weakest link revisited. *IEEE Sec. & Privacy*, 1(2):72–76, Mar 2003.

[4] A. Attar, R. M. Rad, and R. E. Atani. A survey of image spamming and filtering techniques. *Artif. Intell. Rev.*, 40(1):71–105, 2013.

[5] M. Barreno, P. L. Bartlett, F. J. Chi, A. D. Joseph, B. Nelson, B. I. Rubinstein, U. Saini, and J. D. Tygar. Open problems in the security of learning. In *Proc. 1st ACM Workshop on Artificial Intell. and Sec.*, AISec '08, pp. 19–26, 2008. ACM.

[6] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Proc. of the ACM Symp. on Information, Computer and Comm. Sec.*, ASIACCS '06, pp. 16–25, 2006. ACM.

[7] A. Barth, B. I. Rubinstein, M. Sundararajan, J. C. Mitchell, D. Song, and P. L. Bartlett. A learning-based approach to reactive security. *IEEE Trans. on Dependable and Secure Computing*, 9(4):482–493, 2012.

[8] B. Biggio, Z. Akhtar, G. Fumera, G. L. Marcialis, and F. Roli. Security evaluation of biometric authentication systems under real spoofing attacks. *IET Biometrics*, 1(1):11–24, 2012.

[9] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli. Bagging classifiers for fighting poisoning attacks in adversarial environments. In C. Sansone et al., eds., *10th Int'l Workshop on MCSs*, vol. 6713 of *LNCS*, pp. 350–359. Springer, 2011.

[10] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli. Evasion attacks against machine learning at test time. In H. Blockeel et al., eds., *European Conf. on Machine Learning and Principles and Practice of Knowl. Discovery in Databases, Part III*, vol. 8190 of *LNCS*, pp. 387–402. Springer, 2013.

[11] B. Biggio, I. Corona, B. Nelson, B. Rubinstein, D. Maiorca, G. Fumera, G. Giacinto, and F. Roli. Security evaluation of support vector machines in adversarial environments. In Y. Ma and G. Guo, eds., *Support Vector Machines Applications*, pp. 105–153. Springer International Publishing, 2014.

[12] B. Biggio, L. Didaci, G. Fumera, and F. Roli. Poisoning attacks to compromise face templates. In *6th IAPR Int'l Conf. on Biometrics*, pp. 1–7, 2013.

[13] B. Biggio, G. Fumera, I. Pillai, and F. Roli. A survey and experimental evaluation of image spam filtering techniques. *Pattern Rec. Letters*, 32(10):1436 – 1446, 2011.

[14] B. Biggio, G. Fumera, and F. Roli. Adversarial pattern classification using multiple classifiers and randomisation. In *12th Joint IAPR Int'l Workshop on Structural and Syntactic Pattern Rec.*, vol. 5342 of *LNCS*, pp. 500–509, 2008. Springer-Verlag.

[15] B. Biggio, G. Fumera, and F. Roli. Evade hard multiple classifier systems. In O. Okun and G. Valentini, eds., *Supervised and Unsupervised Ensemble Methods and Their Applications*, vol. 245 of *Studies in Computational Intell.*, pp. 15–38. Springer, 2009.

[16] B. Biggio, G. Fumera, and F. Roli. Multiple classifier systems for adversarial classification tasks. In J. A. Benediktsson, J. Kittler, and F. Roli, eds., *Proc. 8th Int'l Workshop on MCSs*, vol. 5519 of *LNCS*, pp. 132–141. Springer, 2009.

[17] B. Biggio, G. Fumera, and F. Roli. Multiple classifier systems for robust classifier design in adversarial environments. *Int'l J. Mach. Learn. Cybern.*, 1(1):27–41, 2010.

[18] B. Biggio, G. Fumera, and F. Roli. Multiple classifier systems under attack. In N. E. Gayar et al., eds., *9th Int'l Workshop on MCSs*, vol. 5997 of *LNCS*, pp. 74–83. Springer, 2010.

[19] B. Biggio, G. Fumera, and F. Roli. Design of robust classifiers for adversarial environments. In *IEEE Int'l Conf. on Systems, Man, and Cybern.*, pp. 977–982, 2011.

[20] B. Biggio, G. Fumera, and F. Roli. Security evaluation of pattern classifiers under attack. *IEEE Trans. on Knowl. and Data Engineering*, 26(4):984–996, 2014.

[21] B. Biggio, G. Fumera, F. Roli, and L. Didaci. Poisoning adaptive biometric systems. In G. Gimel'farb et al., eds., *Structural, Syntactic, and Statistical Pattern Rec.*, vol. 7626 of *LNCS*, pp. 417–425. Springer, 2012.

[22] B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. In J. Langford et al., eds., *29th Int'l Conf. on Mach. Learn.*. Omnipress, 2012.

[23] B. Biggio, I. Pillai, S. R. Bulò, D. Ariu, M. Pelillo, and F. Roli. Is data clustering in adversarial settings secure? In *Proc. Workshop on Artificial Intell. and Security*, AISec '13, pp. 87–98, 2013. ACM.

[24] M. Brückner, C. Kanzow, and T. Scheffer. Static prediction games for adversarial learning problems. *J. Mach. Learn. Res.*, 13:2617–2654, 2012.

[25] M. Brückner and T. Scheffer. Nash equilibria of static prediction games. In Y. Bengio et al., eds., *Advances in Neural Information Processing Systems 22*, pp. 171–179. 2009.

[26] M. Brückner and T. Scheffer. Stackelberg games for adversarial prediction problems. In *17th Int'l Conf. Knowl. Disc. and Data Mining*, KDD '11, pp. 547–555, 2011. ACM.

[27] A. Christmann and I. Steinwart. On robust properties of convex risk minimization methods for pattern recognition. *J. Mach. Learn. Res.*, 5:1007–1034, 2004.

[28] G. F. Cretu, A. Stavrou, M. E. Locasto, S. J. Stolfo, and A. D. Keromytis. Casting out demons: Sanitizing training data for anomaly sensors. In *IEEE Symp. on Security and Privacy*, pp. 81–95, 2008. IEEE CS.

[29] C. Croux, P. Filzmoser, and M. R. Oliveira. Algorithms for projection - pursuit robust principal component analysis. *Chemometrics Intell. Lab. Sys.*, 87(2):218–225, 2007.

[30] G. Cybenko and C. E. Landwehr. Security analytics and measurements. *IEEE Security & Privacy*, 10(3):5–8, 2012.

[31] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *10th Int'l Conf. on Knowl. Disc. and Data Mining*, pp. 99–108, 2004.

[32] O. Dekel, O. Shamir, and L. Xiao. Learning to classify with missing and corrupted features. *Machine Learning*, 81:149–178, 2010.

[33] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

[34] C. Dwork. Differential privacy. In *Proc. 33rd Int'l Colloquium on Automata, Languages and Programming (ICALP)*, pp. 1–12. Springer, 2006.

[35] P. Fogla, M. Sharif, R. Perdisci, O. Kolesnikov, and W. Lee. Polymorphic blending attacks. In *Proc. 15th Conf. on USENIX Security Symp.*, 2006. USENIX Association.

[36] J. Galbally, C. McCool, J. Fierrez, S. Marcel, and J. Ortega-Garcia. On the vulnerability of face verification systems to hill-climbing attacks. *Pattern Recogn.*, 43(3):1027–1038, 2010.

[37] A. Globerson and S. T. Roweis. Nightmare at test time: robust learning by feature deletion. In W. W. Cohen and A. Moore, eds., *Proc. 23rd Int'l Conf. on Mach. Learn.*, vol. 148, pp. 353–360. ACM, 2006.

[38] M. Großhans, C. Sawade, M. Brückner, and T. Scheffer. Bayesian games for adversarial regression problems. In *Proc. 30th Int'l Conf. Mach. Learn.*, vol. 28, 2013.

[39] P. Haider, L. Chiarandini, and U. Brefeld. Discriminative clustering for market segmentation. In *Proc. 18th Int'l Conf. on Knowl. Disc. and Data Mining*, KDD '12, pp. 417–425, 2012. ACM.

[40] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. Probability and Mathematical Statistics. John Wiley and Sons, NY, USA, 1986.

[41] L. Huang, A. D. Joseph, B. Nelson, B. Rubinstein, and J. D. Tygar. Adversarial machine learning. In *4th Workshop on Artificial Intell. and Sec.*, pp. 43–57, 2011.

[42] A. K. Jain, A. Ross, S. Pankanti, and S. Member. Biometrics: A tool for information security. *IEEE Trans. on Information Forensics and Security*, 1:125–143, 2006.

[43] S. Jana and V. Shmatikov. Abusing File Processing in Malware Detectors for Fun and Profit. In *Proc. 33rd IEEE Symp. on Sec. & Privacy*, pp. 80–94, 2012.

[44] A. D. Joseph, P. Laskov, F. Roli, J. D. Tygar, and B. Nelson. Machine Learning Methods for Computer Security (Dagstuhl Perspectives Workshop 12371). *Dagstuhl Manifestos*, 3(1):1–30, 2013.

[45] M. Kantarcioglu, B. Xi, and C. Clifton. Classifier evaluation and attribute selection against active adversaries. *Data Mining and Knowl. Discovery*, pp. 1–45, 2010.

[46] M. Kloft and P. Laskov. Online anomaly detection under adversarial impact. In *Proc. of the 13th Int'l Conf. on Artificial Intell. and Statistics*, pp. 405–412, 2010.

[47] A. Kolcz and C. H. Teo. Feature weighting for improved classifier robustness. In *6th Conf. on Email and Anti-Spam*, 2009.

[48] L. I. Kuncheva. Classifier ensembles for detecting concept change in streaming data: Overview and perspectives. In O. Okun and G. Valentini, eds., *Workshop on Supervised and Unsupervised Ensemble Methods and their Applications*, 2008.

[49] W. Liu and S. Chawla. Mining adversarial patterns via regularized loss minimization. *Machine Learning*, 81(1):69–83, 2010.

[50] D. Lowd and C. Meek. Adversarial learning. In A. Press, ed., *Proc. 11th Int'l Conf. on Knowl. Disc. and Data Mining*, pp. 641–647, 2005.

[51] D. Lowd and C. Meek. Good word attacks on statistical spam filters. In *2nd Conf. on Email and Anti-Spam*, 2005.

[52] D. Maiorca, I. Corona, and G. Giacinto. Looking at the bag is not enough to find the bomb: an evasion of structural methods for malicious pdf files detection. In *Proc. 8th Symp. on Inform., Comp. and Comm. Sec.*, ASIACCS '13, pp. 119–130, 2013. ACM.

[53] R. A. Maronna, R. D. Martin, and V. J. Yohai. *Robust Statistics: Theory and Methods*. Probability and Mathematical Statistics. John Wiley and Sons, NY, USA, 2006.

[54] M. Martinez-Diaz, J. Fierrez, J. Galbally, and J. Ortega-Garcia. An evaluation of indirect attacks and countermeasures in fingerprint verification systems. *Pattern Rec. Letters*, 32(12):1643 – 1651, 2011.

[55] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia. Exploiting machine learning to subvert your spam filter. In *Proc. 1st Workshop on Large-Scale Expl. and Em. Threats*, pp. 1–9, 2008. USENIX.

[56] B. Nelson, B. Biggio, and P. Laskov. Understanding the risk factors of learning in adversarial environments. In *4th Workshop on Artificial Intell. and Sec.*, AISec '11, pp. 87–92, 2011. ACM.

[57] B. Nelson, B. I. Rubinstein, L. Huang, A. D. Joseph, S. J. Lee, S. Rao, and J. D. Tygar. Query strategies for evading convex-inducing classifiers. *J. Mach. Learn. Res.*, 13:1293–1332, 2012.

[58] B. Nelson, B. I. P. Rubinstein, L. Huang, A. D. Joseph, S. hon Lau, S. Lee, S. Rao, A. Tran, and J. D. Tygar. Near-optimal evasion of convex-inducing classifiers. In *Proc. 13th Int'l Conf. on Artificial Intell. and Statistics*, 2010.

[59] J. Newsome, B. Karp, and D. Song. Paragraph: Thwarting signature learning by training maliciously. In *RAID*, LNCS, pp. 81–105. Springer, 2006.

[60] S. Pastrana, A. Orfila, J. E. Tapiador, and P. Peris-Lopez. Randomized anagram revisited. *Journal of Network and Computer Applications*, (0):–, 2014.

[61] R. Perdisci, D. Ariu, and G. Giacinto. Scalable fine-grained behavioral clustering of http-based malware. *Computer Networks*, 57(2):487 – 500, 2013.

[62] N. K. Ratha, J. H. Connell, and R. M. Bolle. An analysis of minutiae matching strength. In J. Bigün and F. Smeraldi, eds., *Int'l Conf. on Audio- and Video-Based Biometric Person Authentication*, vol. 2091 of *LNCS*, pp. 223–228. Springer, 2001.

[63] K. Rieck, P. Trinius, C. Willems, and T. Holz. Automatic analysis of malware behavior using machine learning. *J. Comput. Secur.*, 19(4):639–668, 2011.

[64] R. N. Rodrigues, L. L. Ling, and V. Govindaraju. Robustness of multimodal biometric fusion methods against spoof attacks. *J. Vis. Lang. Comput.*, 20(3):169–179, 2009.

[65] F. Roli, B. Biggio, and G. Fumera. Pattern recognition systems under attack. In J. Ruiz-Shulcloper and G. S. di Baja, eds., *Progress in Pattern Rec., Image Analysis, Computer Vision, and Applications*, vol. 8258 of *LNCS*, pp. 1–8. Springer, 2013.

[66] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. D. Tygar. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proc. 9th Internet Measurement Conf.*, IMC '09, pp. 1–14, 2009. ACM.

[67] B. I. P. Rubinstein, P. L. Bartlett, L. Huang, and N. Taft. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. *J. Privacy and Confidentiality*, 4(1):65–100, 2012.

[68] C. H. Teo, A. Globerson, S. Roweis, and A. Smola. Convex learning with invariances. In J. Platt, D. Koller, Y. Singer, and S. Roweis, eds., *Advances in Neural Information Processing Systems 20*, pp. 1489–1496. MIT Press, Cambridge, MA, 2008.

[69] M. Torkamani and D. Lowd. Convex adversarial collective classification. In *Proc. 30th Int'l Conf. on Mach. Learning*, vol. 28 of *JMLR Proc.*, pp. 642–650. JMLR.org, 2013.

[70] G. L. Wittel and S. F. Wu. On attacking statistical spam filters. In *First Conf. on Email and Anti-Spam*, 2004.

[71] M. Wooldridge. Does game theory work? *IEEE Intell. Systems*, 27(6):76–80, 2012.

[72] H. Xu, C. Caramanis, and S. Mannor. Robustness and regularization of support vector machines. *J. Mach. Learn. Res.*, 10:1485–1510, 2009.

[73] Y. Zhou, M. Kantarcioglu, and B. Thuraisingham. Sparse bayesian adversarial learning using relevance vector machine ensembles. In *IEEE 12th Int'l Conf. Data Mining*, pp. 1206–1211, 2012.

[74] Y. Zhou, M. Kantarcioglu, B. Thuraisingham, and B. Xi. Adversarial support vector machine learning. In *Proc. 18th Int'l Conf. Knowl. Disc. and Data Mining*, KDD '12, pp. 1059–1067, 2012. ACM.

Battista Biggio received the M. Sc. degree in Electronic Eng., with honors, and the Ph. D. in Electronic Eng. and Computer Science, respectively in 2006 and 2010, from the University of Cagliari, Italy. Since 2007 he has been working for the Dept. of Electrical and Electronic Eng. of the same University, where he holds now a postdoctoral position. From May 12th, 2011 to November 12th, 2011, he visited the University of Tübingen, Germany, and worked on

the security of machine learning algorithms to contamination of training data. His research interests currently include: secure / robust machine learning and pattern recognition methods, multiple classifier systems, kernel methods, biometric authentication, spam filtering, and computer security. He serves as a reviewer for several international conferences and journals, including Pattern Recognition and Pattern Recognition Letters. Dr. Biggio is a member of the IEEE Institute of Electrical and Electronics Engineers (Computer Society), and of the Italian Group of Italian Researchers in Pattern Recognition (GIRPR), affiliated to the International Association for Pattern Recognition.

Giorgio Fumera received the M. Sc. degree in Electronic Eng., with honors, and the Ph.D. degree in Electronic Eng. and Computer Science, respectively in 1997 and 2002, from the University of Cagliari, Italy. Since February 2010 he is Associate Professor of Computer Eng. at the Dept. of Electrical and Electronic Eng. of the same University. His research interests are related to methodologies and applications of statistical pattern recognition, and include multiple classifier systems, classification with the reject option, adversarial classification and document categorization. On these topics he published more than sixty papers in international journal and conferences. He acts as reviewer for the main international journals in this field, including IEEE Trans. on Pattern Analysis and Machine Intelligence, Journal of Machine Learning Research, and Pattern Recognition. Dr. Fumera is a member of the IEEE Institute of Electrical and Electronics Engineers (Computer Society), of the Italian Association for Artificial Intelligence (AI*IA), and of the Italian Group of Italian Researchers in Pattern Recognition (GIRPR), affiliated to the International Association for Pattern Recognition.

Fabio Roli received his M. Sc. degree, with honors, and Ph. D. degree in Electronic Eng. from the University of Genoa, Italy. He was a member of the research group on Image Processing and Understanding of the University of Genoa, Italy, from 1988 to 1994. He was adjunct professor at the University of Trento, Italy, in 1993 and 1994. In 1995, he joined the Dept. of Electrical and Electronic Eng. of the University of Cagliari, Italy, where he is now professor of computer engineering and head of the research group on pattern recognition and applications. His research activity is focused on the design of pattern recognition systems and their applications to biometric personal identification, multimedia text categorization, and computer security. On these topics, he has published more than two hundred papers at conferences and on journals. He was a very active organizer of international conferences and workshops, and established the popular workshop series on multiple classifier systems. Dr. Roli is a member of the governing boards of the International Association for Pattern Recognition and of the IEEE Systems, Man and Cybernetics Society. He is Fellow of the IEEE, and Fellow of the International Association for Pattern Recognition.