Key-recovery Attacks on KIDS, a Keyed Anomaly Detection System

Juan E. Tapiador, Agustin Orfila, Arturo Ribagorda, Benjamin Ramos

Abstract—Most anomaly detection systems rely on machine learning algorithms to derive a model of normality that is later used to detect suspicious events. Some works conducted over the last years have pointed out that such algorithms are generally susceptible to deception, notably in the form of attacks carefully constructed to evade detection. Various learning schemes have been proposed to overcome this weakness. One such system is KIDS (Keyed IDS), introduced at DIMVA'10. KIDS' core idea is akin to the functioning of some cryptographic primitives, namely to introduce a secret element (the key) into the scheme so that some operations are infeasible without knowing it. In KIDS the learned model and the computation of the anomaly score are both key-dependent, a fact which presumably prevents an attacker from creating evasion attacks. In this work we show that recovering the key is extremely simple provided that the attacker can interact with KIDS and get feedback about probing requests. We present realistic attacks for two different adversarial settings and show that recovering the key requires only a small amount of queries, which indicates that KIDS does not meet the claimed security properties. We finally revisit KIDS' central idea and provide heuristic arguments about its suitability and limitations.

Index Terms—Adversarial Classification, Anomaly Detection, Intrusion Detection Systems, Secure Machine Learning.

1 Introduction

Many computer security problems can be essentially reduced to separating malicious from non-malicious activities. This is, for example, the case of spam filtering, intrusion detection, or the identification of fraudulent behavior. But, in general, defining in a precise and computationally useful way what is harmless or what is offensive is often too complex. To overcome these difficulties, most solutions to such problems have traditionally adopted a machine-learning approach, notably through the use of classifiers to automatically derive models of (good and/or bad) behavior that are later used to recognize the occurrence of potentially dangerous events.

Recent work (see, e.g., [1], [2] for an overview) has accurately pointed out that security problems differ from other application domains of machine learning in, at least, one fundamental feature: the presence of an adversary who can strategically *play* against the algorithm to accomplish his goals. Thus for example, one major objective for the attacker is to avoid detection. Evasion attacks exploit weaknesses in the underlying classifiers, which are often unable to identify a malicious sample that has been conveniently modified so as to look normal. Examples of such attacks abound. For instance, spammers regularly obfuscate their emails in various ways to avoid detection, e.g. by modifying words that are usually found in spam, or by including a large number of words that do not (e.g., [8], [23]). Similarly, malware and other pieces of attack code can be carefully adapted so as to evade Intrusion Detection Systems (IDS) without compromising the functionality of the attack (see, e.g., [6], [9]).

A few detection schemes proposed over the last few years have attempted to incorporate defenses against evasion attacks. One such system is KIDS (Keyed Intrusion Detection System) [12], introduced by Mrdovic and Drazenovic at DIMVA'10. KIDS is an application-layer network anomaly detection system that extracts a number of features ("words") from each payload. The system then builds a model of normality based both on the frequency of observed features and their relative positions in the payload. KIDS' core idea to impede evasion attacks is to incorporate the notion of a "key", this being a secret element used to determine how classification features are extracted from the payload. The security argument here is simple: even though the learning and testing algorithms are public, an adversary who is not in possession of the key will not know exactly how a request will be processed and, consequently, will not be able to design attacks that thwart detection.

Strictly speaking, KIDS' idea of "learning with a secret" is not entirely new: Wang et al. introduced in [22] Anagram, another payload-based anomaly detection system that addresses the evasion problem in quite a similar manner. We distinguish here between two broad classes of classifiers that use a key. In the first group, that we term *randomized* classifiers, the classifier is entirely public (or, equivalently, is trained with public information only). However, in detection mode some parameters (the key) are randomly chosen every time an instance has to be classified, thus making uncertain for the attacker how the instance will be processed. Note

• *The authors are with the Department of Computer Science, Universidad Carlos III de Madrid, 28911 Leganes, Madrid, Spain.*
E-mail: jestevez@inf.uc3m.es (J.E. Tapiador), adiaz@inf.uc3m.es (A. Orfila), arturo@inf.uc3m.es (A. Ribagorda), benja1@inf.uc3m.es (B. Ramos).

that, in this case, the same instance will be processed differently every time if the key is randomly chosen. We emphasize that randomization can also be applied at training time, although it may only be sufficiently effective when used during testing, at least as far as evasion attacks are concerned. KIDS belongs to a second group, that we call *keyed* classifiers. In this case, there is one secret and persistent key that is used during a period of time, possibly because changing the key implies retraining the classifier. If Kerckhoffs' principle is to be followed, it must be assumed that the security of the scheme depends solely on the secrecy of the key and the procedure used to generate it. Anagram can be used both as randomized or as a keyed classifier, depending on the variant used. We will further discuss this later in Section 6.

1.1 Contributions

In this work, we make the following contributions:

1) We argue that any *keyed* anomaly detection system (or, more generally, any keyed classifier) must preserve one fundamental property: The impossibility for an attacker to recover the key under any reasonable adversarial model. We deliberately choose not to analyze how difficult is for an attacker to evade detection if the classifier is keyed. We believe that this is a related, but different problem.
2) We pose the key-recovery problem as one of adversarial learning. By adapting the adversarial setting proposed by Lowd and Meek [10] in a related problem (revese engineering of a classifier), we introduce the notion of gray- and black-box key-recovery attacks.
3) We present two instantiations of such attacks for KIDS, one for each model. Our attacks take the form of query strategies that make the classifier leak some information about the key. Both are very efficient and show that KIDS does not meet the fundamental security property discussed above. Furthermore, we have implemented and experimentally confirmed the correctness of our attacks.
4) Building on related work in the broader field of secure machine learning (e.g., [1], [2], [3], [5], [10], [13], [14], [15]), we pose some additional questions and provide constructive discussion about the suitability, limitations, and possible structure of keyed classifiers.

The remainder of this paper is organized as follows. In Section 2 we provide a brief overview of related work in the field of adversarial machine learning. For completeness, a description of KIDS is given in Section 3. In Section 4 we introduce the adversarial model adopted, describe and analyze our attacks, and discuss the results obtained experimentally. KIDS's core idea is revisited and further discussed in Section 5, and Section 6 concludes the paper.

2 Related Work

2.1 Classifier Evasion and Adversarial Learning

Dalvi et al. explored in [5] the problem of computing optimal strategies to modify an attack so that it evades detection by a Naïve Bayes classifier. They formulate the problem in game-theoretic terms, where each modification made to an instance comes at a price, and successful detection and evasion have measurable utilities to the classifier and the adversary, respectively. The authors study how to detect such optimally modified instances by adapting the decision surface of the classifier, and also discuss how the adversary might react to this.

The setting used in [5] assumes an adversary with full knowledge of the classifier to be evaded. Shortly after, Lowd and Meek [10] studied how evasion can be done when such information is unavailable. They formulate the adversarial classifier reverse engineering problem (ACRE) as the task of learning sufficient information about a classifier to construct attacks, instead of looking for optimal strategies. The authors use a *membership oracle* as implicit adversarial model: the attacker is given the opportunity to query the classifier with any chosen instance to determine whether it is labeled as malicious or not. Consequently, a reasonable objective is to find instances that evade detection with an affordable number of queries. A classifier is said to be ACRE learnable if there exists an algorithm that finds a minimal-cost instance evading detection using only polynomially-many queries. Similarly, a classifier is ACRE k-learnable if the cost is not minimal but bounded by k. Among the results given in [10], it is proved that linear classifiers with continuous features are ACRE k-learnable under linear cost functions. Therefore, these classifiers should not be used in adversarial environments. Subsequent work by Nelson et al. [14], [15] generalizes these results to convex-inducing classifiers, showing that it is generally not necessary to reverse engineer the decision boundary to construct undetected instances of near-minimal cost.

For the interested reader, Nelson et al. [13] have recently surveyed some open problems and challenges related to the classifier evasion problem. More generally, some additional works have revisited the role of machine learning in security applications, with particular emphasis on anomaly detection [7], [17], [18], [19].

2.2 Strategies to Thwart Evasion

Kolesnikov et al. [9] demonstrate that polymorphic mimicry worms, based on encryption and data encoding to obfuscate their content, are able to evade frequency distribution-based anomaly detectors like PAYL [21]. PAYL models byte-value frequency distributions (i.e., 1-grams), so detection can be avoided by padding anomalous sequences with an appropriate amount of normal traffic. In order to counteract polymorphic mimicry worms, PAYL authors developed Anagram [22], an anomaly detector that models n-grams observed in normal traffic. Anagram also introduces a new strategy,

called randomization, to hinder evasion. There are two possible kinds of randomization, namely randomized modeling and randomized testing. In the former, packets are split into several substrings using a randomly-generated bitmask. Substrings coming from the same packet position are modeled and tested separately. Since the bitmask is kept secret, an attacker only succeeds if he manages to craft an attack vector such that the data is normal with respect to any randomly selected portion of a packet. This clearly makes evasion harder, but substantially increases the overhead of the IDS. Alternatively, randomized testing also partitions packets randomly into several chunks, but tests each of them against the same classifier, which does not incur any substantial overhead.

Randomization and/or using an ensemble of classifiers have also been proposed in the context of spam detection. For example, Biggio et al. [3] studied how to introduce randomness in the design of the classifier, preventing the adversary from having exact knowledge about one or more system parameters. A similar approach was presented by Perdisci et al. in [16]. The work in [3] uses multiple classifiers and randomly chooses the weights assigned to each classifier in the decision. The task for the attacker is much harder then, since he can never guess the detector's configuration. The main problem of this strategy is that it can influence negatively the overall detection performance, particularly increasing the false positive rate.

Zhou et al. [23] presented similar strategies to thwart good-word attacks on spam filters. Their scheme transforms each email into a bag of multiple segments (instances), and then applies multiple-instance logistic regression to the bags. An email is classified as spam if at least one instance in the corresponding bag is spam; otherwise it is marked as legitimate. This bags-of-words strategy performs better than single-instance learners such as Support Vector Machines (SVMs) or Naïve Bayes. A similar approach was explored in [20] to detect masquerade mimicry attacks.

2.3 Towards Secure Machine Learning

Barreno et al. [1], [2] have pondered on the risks of applying machine learning algorithms to security domains. They introduce a taxonomy that groups attacks on machine learning systems into different categories, depending on whether the adversary influences training or just analyzes an already trained system; whether the goal is to force just one misclassification, or else to generate too many so the system becomes unusable; etc. The authors also provide useful discussion on potential countermeasures and enumerate various open problems.

3 KIDS – A Keyed Intrusion Detection System

In 2010, Mrdovic and Drazenovic [12] proposed *Keyed Intrusion Detection System* (KIDS), a key dependent network anomaly detector that inspects packet payloads. The proposal tries to adapt to intrusion detection systems Kerckhoffs' principle stating that a cryptosystem should be secure even if everything about the system, except the key, is public knowledge.

3.1 Training mode

KIDS divides each payload into *words*. A word is defined as a sequence of bytes located between two *delimiters*, these being any two special bytes belonging to a secret set D. A key D consists therefore of a chosen set of delimiters. Each key produces a unique set of normal words and, accordingly, a unique classifier.

KIDS is trained using normal (i.e., attack-free) payloads only. Given a key, each payload in the training set is segmented into words and the frequency of each word is counted. In addition, the number of occurrences of pairs of words (called transitions) is also counted. The model consists of these two lists: one with each observed word, w_i, and its frequency, $n(w_i)$; and another with each observed transition, $w_i \rightarrow w_j$, and its frequency, $n(w_i \rightarrow w_j)$.

3.2 Detection mode

In the detection phase, KIDS assigns an anomaly score, $S(p)$, to each incoming payload p. Subsequently, p is labeled as anomalous if $S(p) > \tau$, where τ is a conveniently chosen threshold.

The anomaly score is given by the product of two separate scores. The first, termed the word score and denoted $S_w(p)$, is computed as:

$$S_w(p) = \frac{1}{k} \sum_{i=1}^{k} \frac{1}{n(w_i)} \tag{1}$$

where k is the number of words in p and $n(w_i)$ the number of appearances of w_i, as computed during training. If a word w_i that did not appear during training appears in p (i.e., $n(w_i) = 0$), the corresponding term in the sum is set to 2 instead of infinity. Thus, every previously unseen word contributes twice to $S_w(p)$ compared to a word that was seen once ($n(w_i) = 1$).

The transition score, denoted $S_t(p)$, is calculated according to a similar formula:

$$S_t(p) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{n(t_i)} \tag{2}$$

where m is the number of transitions in p (i.e., $k - 1$) and $n(t_i)$ is the frequency of transition t_i in the learned model.

The overall score $S(p)$ assigned to a payload is obtained as:

$$S(p) = S_w(p) * S_t(p) \tag{3}$$

Thus, the appearance of frequent words and transitions contributes to mantain $S(p)$ low, and vice versa.

3.3 Key selection

Keys in KIDS are selected so as to ensure good detection quality. The ROC (Receiver Operating Characteristic) curve is chosen in [12] as the method to quantify how well a particular key performs. The authors employ a labeled dataset consisting of attack-free HTTP traffic and tailored attacks generated with Metasploit [11]. An initial key composed of 20 delimiters (CR, LF, TAB, SPACE, ',', '.', ':', '/', ';', '&', '?', '=', '(', ')', '[', ']', '"', ';', '<', '>') was first selected using domain-specific knowledge, and the obtained ROC curve shows the model thus built is quite effective.

The authors explored next whether similar results can be obtained using random keys. Different keys of size 15, 20, 25, and 30 were generated by choosing random delimiters with values between 0 and 255. According to their experimental results, some of these random keys yield, in terms of ROC curves, detection results as good as those obtained with the human-generated key. The paper suggests to repeat this procedure every time a new key has to be chosen.

4 Key-recovery Attacks on KIDS

In this section we describe various attacks on KIDS aimed at recovering the secret set of delimiters (i.e., the key). We group these attacks into two broad classes, depending on what feedback from KIDS the attacker may have access to. Before presenting our attacks, we first describe the adversarial model adopted and give grounds for our main assumptions.

4.1 Adversarial Model and Notation

When assessing the security of systems such as KIDS, one major problem comes from the absence of widely accepted adversarial models giving a precise description of the attacker's goals and his capabilities. Barreno et al. [2] have recently introduced one such model for secure machine learning and discussed various general attack categories. Our work does not fit well within Barreno et al.'s model because our main goal is not to attack the learning algorithm itself, but to recover one piece of secret information that, subsequently, may be essential to succesfully launch an evasion attack. In some respects, our work is far more similar to that of Lowd and Meek [10], where the focus is on the role of active experimentation with a classifier. In such a scenario, it is absolutely essential for the attacker to be able to: (1) send queries to the classifier; and (2) get some feedback about properties of the query as processed by the system. We emphasize that the ability to do this is close to the bare minimum required to analyze the security of any scheme. In our case, our central assumption is given next.

Assumption 1: The attacker can interact with KIDS by providing some chosen input (i.e., a payload) and observing the outcome. We distinguish two cases here, depending on what sort of output information from KIDS the attacker has access to. In a *gray-box model*, we assume the anomaly score is observable by the attacker. Alternatively, we refer to a *black-box model* when the adversary has access only to the binary label normal/anomalous given to the input payload.

We believe that both models could be realistic for a variety of scenarios. Firstly, the ability to feed the IDS with inputs is available to everyone who can access the service protected by the IDS. Thus for example, such queries would be arbitrarily chosen payloads sent to an HTTP, FTP, SQL, etc. server.

Getting feedback from the IDS seems a priori more problematic, but it would be unsafe to assume that this knowledge is unavailable to the attacker. In the case of the black-box model, one potential scenario involves an attacker who can determine whether an alarm has been generated or not. This information could be obtained by observing the network and checking if an alarm is sent to the security officer, either directly by observing the channel or indirectly through some side channels. If the attacker is an insider, even one with few privileges, obaining this information may be easier. The gray-box model is stronger in the sense that getting access to the anomaly score seems rather unrealistic. Apart from the merely theoretical interest, we believe that the score may be also obtained by the attacker if, for example, such a value is included in the alarm sent to the security officer. Some real-world IDS do this in order to provide the decision maker with as much information as possible about the potential attack. Thus, if such alarms are not encrypted, an observer could get access to the score.

A related question is that attacking the system involves sending numerous payloads to it, many of which will generate alarms. This may obviously raise suspicions, so the attacker must be careful, e.g., by spreading them over a period of time. Alternatively, the attacker may be given the ability to block the alerts during some time. This will be enough if the time the attacker needs to recover the key is sufficiently low, which will depend on the overall complexity of the attack.

In any case, we subscribe to the generally accepted philosophy [2]: *"While we think that this is the most accurate assumption for most cases, if we do err, we wish to follow the common practice in computer security research of erring on the side of overestimating the attacker's capabilities rather than underestimating them."*

For clarity, we summarize in Fig. 1 the notation used in the remainder of this paper.

4.2 Key-recovery on Gray-box KIDS

In this attack we assume the attacker has access to the anomaly score assigned to a chosen payload. Furthermore, it is reasonable to assume that some normal payloads are known too. (Consider, for example, the case of an IDS analyzing HTTP requests sent to a publicly accessible web server, where a large number of such payloads will be known by the attacker.) Let p be

Symbol	Meaning
w_i, ξ_i	Words
$w_i \parallel w_j$	Concatenation
$w_i \rightarrow w_j$	Transition from w_i to w_j
d, d_i	Delimiters
$D = \{d_1, \ldots, d_{\lvert D \rvert}\}$	Secret key (delimiters)
p, q, r, t	Payloads
$p[i]$	i-th byte of payload p
$n(x)$	Number of occurences of x
$S_w(p)$	Word score for p
$S_t(p)$	Transition score for p
$S(p)$	Overall score for p
α	S_t value assigned to a payload with no transitions
β	Frequency assigned to a word/transition unseen during training
$\texttt{anom}(p)$	Anomaly label (`true` or `false`) given to p

Fig. 1. Notation used.

one such normal payload. A straightforward strategy to identify what elements of p belong to the key D consists of feeding KIDS with the first byte of p, then with the first two bytes of p, and so on. When the next-to-the-last byte happens to be a delimiter, KIDS will detect a transition where the left word is likely to have been seen during training, whereas the right word is often unknown (since it is truncated). At this point, the anomaly score will suffer a slight decrement. By conveniently repeating the procedure, all the delimiters present in p can be recovered.

Regardless of the technical details, the main drawback of the naïve strategy discussed above is that the attacker will only be able to recover those key elements present in the normal payloads available, which may well be just a fraction of all of them. Besides, the complexity of such an attack is linear in the number of payloads and their lengths. We next describe a different approach that obtains *all* the key elements more efficiently and without directly relying on normal payloads.

The attack works by constructing a probing payload as follows. Let $p = w_1 \parallel d \parallel w_2$, where:

(i) $n(w_1) > 0$
(ii) $n(w_2) = 0$
(iii) $n(w_1 \parallel d \parallel w_2) = 0$

Finding such w_1 and w_2 is not difficult following the procedure discussed above. The technical details (i.e., how to detect a word by analyzing changes in the anomaly score) will be clear after this section and are also provided in Section 4.4.

We now feed KIDS with p and observe the resulting anomaly score. There are two cases, depending on whether d is part of the key D or not:

- **Case 1:** $d \notin D$
  In such a case, p is processed as just one word, which in turn has not been previously seen as $n(w_1 \parallel d \parallel w_2) = 0$. Consequently, we have

$$S_w(p) = \beta \tag{4}$$

  where β is the value assigned to a previously unseen word or transition. Even though in KIDS this value is set to 2, in our analysis we consider the more general case. Likewise

$$S_t(p) = \alpha \tag{5}$$

  where α is the S_t value given by KIDS to a payload containing just one word and, therefore, no transitions. Even though such a case is not discussed in [12], the intuition dictates that either $\alpha = 1$ (and therefore the transition score does not have any influence on the overall score), or else $\alpha > 1$ (possibly with $\alpha = 2$, in order to be consistent with the rationale about β) and it is considered a "transition" unseen during training. In any case, the overall anomaly score would be

$$S(p) = \beta\alpha \tag{6}$$

- **Case 2:** $d \in D$
  In this case, p is split into two words, w_1 and w_2. Thus we have

$$S_w(p) = \frac{1}{2}\left(\frac{1}{n(w_1)} + \beta\right) \tag{7}$$

  and

$$S_t(p) = \beta \tag{8}$$

  since, by construction, $n(w_1) > 0$ and $n(w_2) = 0$, so therefore no transition $w_1 \rightarrow w_2$ could have been seen during training. Therefore, the overall score given to p is

$$S(p) = \beta\left[\frac{1}{2}\left(\frac{1}{n(w_1)} + \beta\right)\right] \tag{9}$$

Now expressions (6) and (9) can be used to analyze $S(p)$ and tell whether d is part of the key or not. Assume that the attacker repeats the procedure for each possible value of d. Since α, β and $n(w_1)$ are constant values, the 256 possible values of d are split into two sets: those producing an anomaly score of $\beta\alpha$ and those for which the result is $\beta\left[\frac{1}{2}\left(\frac{1}{n(w_1)}+\beta\right)\right]$. Note that both values will only be equal if $\alpha = \frac{1}{2}\left(\frac{1}{n(w_1)} + \beta\right)$, which is extremely unlikely as w_1 is unknown to the defender. Furthermore, even if the defender chooses α and β so as to force this equality to hold, it has to be done only for one particular w_1 that, besides, is unknown at the time of the attack.

Obviously, the attacker does not know the concrete values of (6) and (9). However, he can group together all the delimiters d that produce the same score, obtaining the sets D_1 and D_2. Note that these two sets form a partition of the set of delimiters, one of them being the complete key. Determining which one is the key is now

Gray-box Key Recovery

Input:
w_1, w_2 such that $n(w_1) > 0$, $n(w_2) = 0$
Algorithm:
1. $D_1 \leftarrow \emptyset$
2. $D_2 \leftarrow \emptyset$
3. **for** $d = 0$ **to** 255 **do**
4. $p \leftarrow (w_1 \parallel d \parallel w_2)$
5. **if** $S(p) = S(w_1 \parallel \hat{d} \parallel w_2)\ \forall \hat{d} \in D_1$ **then**
6. $D_1 \leftarrow D_1 \cup \{d\}$
7. **else**
8. $D_2 \leftarrow D_2 \cup \{d\}$
9. **end-if**
10. **end-for**
11. $q \leftarrow w_2$
12. **if** $S(q) = S(w_1 \parallel \hat{d} \parallel w_2)\ \forall \hat{d} \in D_1$ **then**
13. **return** D_2
14. **else**
15. **return** D_1

Fig. 2. Key-recovery attack on gray-box KIDS.

easy. Assume that the attacker now queries KIDS with a payload $q = w_2$, which will be assigned the score $S(q) = \beta\alpha$. Now, if the delimiters in D_1 were assigned exactly the same score, i.e., $S(q)$, then the key is the set D_2. Otherwise, the key is D_1. We make a final remark on the impossibility of repairing the scheme by using concrete values of α and β, as exactly the same attack can be applied no matter what constants are used.

The overall key-recovery attack is summarized in the algorithm given in Fig. 2.

4.2.1 Complexity

The attack makes exactly 257 queries to KIDS: 256 with each tentative key element d, plus one final query to determine which subset corresponds to the key. It is worth noting that the attack *always* takes 257 queries, regardless of the key size $|D|$. In other words, the key is not recovered by checking all the $\binom{256}{|D|} = \frac{256!}{|D|!(256-|D|)!}$ possible keys, but rather all the possible constituent elements one by one.

Later in Section 4.4 we give procedures to get words w_1 and w_2 for settings where the attacker does not know them. Obtaining such words incurs a few additional queries to KIDS.

4.3 Key-recovery on Black-box KIDS

In this section we present a key-recovery attack when the only information about a payload an adversary gets from KIDS is its classification label, i.e., whether it is normal or anomalous. In some respects, this information is less fine-grained than the anomaly score, so it is reasonable to expect that attacks working under this assumption will be slightly more complex.

The central idea behind our attack is actually quite simple. We will provide KIDS with a normal payload concatenated with a carefully constructed tail. Such a tail contains a large number of unseen words separated by the candidate delimiter. If the delimiter does not belong to the key, the entire tail will be processed as just one word and the anomaly score will be roughly similar to that of the original payload. If this is the case, then the payload will be marked as normal with high probability. Conversely, if the delimiter does belong to the key, the tail will be fragmented into a large number of previously unseen words and transitions. This will negatively impact the anomaly score, invariably resulting in an anomalous payload. We next provide a more formal description and analysis of the attack.

Assume a payload q composed of words $\xi_1, \ldots, \xi_k$ separated by delimiters $d_{j_1}, \ldots, d_{j_{k-1}}$, i.e., $q = \xi_1 \parallel d_{j_1} \parallel \xi_2 \parallel d_{j_2} \parallel \cdots \parallel d_{j_{k-1}} \parallel \xi_k$. Assume too that q is normal, i.e. $\texttt{anom}(q) = \texttt{false}$. Let w_2 be a word unseen during training, i.e., $n(w_2) = 0$. We now construct a probing payload p consisting of payload q followed by a tail t, where t is formed by the concatenation of ℓ repetitions of w_2 separated by the candidate delimiter d; i.e., $t = d \parallel w_2 \parallel d \parallel w_2 \parallel d \parallel \cdots \parallel d \parallel w_2$ and $p = q \parallel t$.

We next analyze the behavior of KIDS when p is provided as input. Again, there are two cases, depending on whether d is part of the key D or not:

- **Case 1:** $d \notin D$
  In this case, p is split into k words: the first $k-1$ original words already present in q plus the tail t preceded by ξ_n. Thus, we have

$$\begin{aligned} S_w(p) &= \frac{1}{k}\left(\sum_{i=1}^{k-1}\frac{1}{n(\xi_i)} + \frac{1}{n(\xi_k \parallel t)}\right) \\ &= \frac{1}{k}\left(\sum_{i=1}^{k-1}\frac{1}{n(\xi_i)} + \beta\right) \end{aligned} \tag{10}$$

  We also have

$$S_w(q) = \frac{1}{k}\sum_{i=1}^{k}\frac{1}{n(\xi_i)} \tag{11}$$

  Now using (11), expression (10) can be rewritten as

$$\begin{aligned} S_w(p) &= \frac{1}{k}\left(kS_w(q) - \frac{1}{n(\xi_k)} + \beta\right) \\ &= S_w(q) + \frac{1}{k}\left(\beta - \frac{1}{n(\xi_k)}\right) \end{aligned} \tag{12}$$

  Similarly, for the transition score we have

$$\begin{aligned} S_t(p) &= \frac{1}{k-1}\left(\sum_{i=1}^{k-2}\frac{1}{n(\xi_i \rightarrow \xi_{i+1})} + \right. \\ &\quad \left. \frac{1}{n(\xi_{k-1} \rightarrow (\xi_k \parallel t))}\right) \\ &= \frac{1}{k-1}\left(\sum_{i=1}^{k-2}\frac{1}{n(\xi_i \rightarrow \xi_{i+1})} + \beta\right) \end{aligned} \tag{13}$$

Again, (13) can be expressed in terms of $S_t(q)$ as

$$\begin{aligned} S_t(p) = & \frac{1}{k-1}\Bigg((k-1)S_t(q) - \\ & \frac{1}{n(\xi_{k-1} \to \xi_k)} + \beta\Bigg) \\ = & S_t(q) + \frac{1}{k-1}\left(\beta - \frac{1}{n(\xi_{k-1} \to \xi_k)}\right) \end{aligned} \tag{14}$$

Note that, in both (12) and (14), the only difference with respect to $S_w(q)$ and $S_t(q)$ is the addition of a positive term. For convenience, let us call them

$$\Delta_w = \frac{1}{k}\left(\beta - \frac{1}{n(\xi_k)}\right) \tag{15}$$

and

$$\Delta_t = \frac{1}{k-1}\left(\beta - \frac{1}{n(\xi_{k-1} \to \xi_k)}\right) \tag{16}$$

Thus we have $S_w(p) = S_w(q) + \Delta_w$ and $S_t(p) = S_t(q) + \Delta_t$. The resulting anomaly score is therefore

$$\begin{aligned} S(p) = & S_w(p)S_t(p) \\ = & (S_w(q) + \Delta_w)(S_t(q) + \Delta_t) \\ = & S_w(q)S_t(q) + S_w(q)\Delta_t + S_t(q)\Delta_w + \Delta_w\Delta_t \\ = & S(q) + (S_w(q)\Delta_t + S_t(q)\Delta_w + \Delta_w\Delta_t) \\ = & S(q) + \Delta \end{aligned} \tag{17}$$

The right-hand side term Δ in (17) depends on k and q's anomaly score. An upper bound for its contribution to p's anomaly score can be derived as follows. On the one hand

$$S_w(q) < \beta \qquad \text{and} \qquad S_t(q) < \beta \tag{18}$$

Note, however, that q is normal and therefore both scores will be significantly lower than β. On the other hand

$$\Delta_w < \frac{\beta}{k} \qquad \text{and} \qquad \Delta_t < \frac{\beta}{k-1} \tag{19}$$

Thus

$$\begin{aligned} \Delta = & S_w(q)\Delta_t + S_t(q)\Delta_w + \Delta_w\Delta_t \\ < & \frac{\beta^2}{k} + \frac{\beta^2}{k-1} + \frac{\beta^2}{k(k-1)} \\ < & \frac{\beta^2}{k-1} + \frac{\beta^2}{k-1} + \frac{\beta^2}{k-1} = \frac{3\beta^2}{k-1} \end{aligned} \tag{20}$$

Recall, that $\texttt{anom}(p) = \texttt{false}$ iff $S(p) = S(q) + \Delta < \tau$. As the increment Δ in q's anomaly score can be upper bounded by (20), the probability of p being classified as normal essentially depends on the following condiditons:

(i) q is "sufficiently" normal, i.e. $S(q)$ is very low.
(ii) k is "sufficiently" large, i.e. Δ is very low.

In Fig. 3 we give plots of the upper bound for Δ in relation to the detection threshold τ. For example, if $\tau = 1.5$ and q is tokenized into $k = 40$ words, Δ will be at most 21% of τ (i.e., $S(p) < S(q) + 0.21\tau$). Consequently, this means that p will be classified as normal if $S(q) \leq 0.79\tau$.

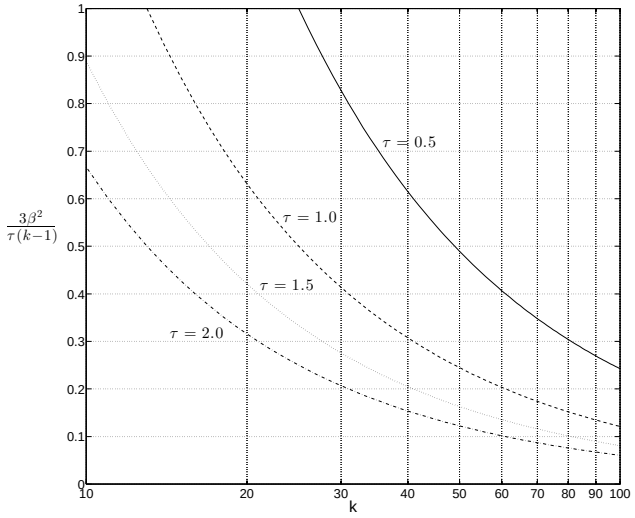

Fig. 3. Upper bounds of the anomaly score increment as a fraction of τ when $d \notin D$ in the black-box attack ($\beta = 2$).

Note that in this scenario the attacker has no control over the internal structure of q, as the key D is unknown and, therefore, k is unknown too. Consequently, success is likely but not guaranteed, a fact which introduces a probabilistic component in the attack. We will address this point later on when discussing the overall procedure. Nevertheless, we suggest to use a payload q as long and frequent as possible, as this will increase the likelihood of satisfying at least one of the previous conditions. Furthermore, the probability of success can be increased by using a q formed by the concatenation of various normal payloads. This will translate into a slight increment of the score due to potentially anomalous transitions in the limits between the original payloads, but will considerably increase k.

- **Case 2:** $d \in D$
  In this case, p is split into $k + \ell$ words: the first k original words already present in q plus ℓ times the word w_2. Thus, we have

$$\begin{aligned} S_w(p) = & \frac{1}{k+\ell}\left(\sum_{i=1}^{k}\frac{1}{n(\xi_i)} + \sum_{i=1}^{\ell}\frac{1}{n(w_2)}\right) \\ = & \frac{1}{k+\ell}\left(\sum_{i=1}^{k}\frac{1}{n(\xi_i)} + \ell\beta\right) \end{aligned} \tag{21}$$

Again, this can be rewritten in terms of $S_w(q)$ as

$$\begin{aligned} S_w(p) = & \frac{1}{k+\ell}(kS_w(q) + \ell\beta) \\ = & \frac{k}{k+\ell}S_w(q) + \frac{\ell\beta}{k+\ell} \end{aligned} \tag{22}$$

Similarly, the transition score is given by

$$\begin{aligned} S_t(p) = & \frac{1}{k+\ell-1}\Bigg(\sum_{i=1}^{k-1}\frac{1}{n(\xi_i \to \xi_{i+1})} + \\ & \frac{1}{n(\xi_n \to w_2)} + \sum_{i=1}^{\ell-1}\frac{1}{n(w_2 \to w_2)}\Bigg) \\ = & \frac{1}{k+\ell-1}\Bigg(\sum_{i=1}^{k-1}\frac{1}{n(\xi_i \to \xi_{i+1})} + \ell\beta\Bigg) \end{aligned} \tag{23}$$

which can be rewritten in terms of $S_t(q)$ as

$$\begin{aligned} S_t(p) = & \frac{1}{k+\ell-1}\Big((k-1)S_t(q) + \ell\beta\Big) \\ = & \frac{k-1}{k+\ell-1}S_t(q) + \frac{\ell\beta}{k+\ell-1} \end{aligned} \tag{24}$$

Note that, in both (22) and (24), the terms multiplying $S_w(q)$ and $S_t(q)$ tend to 0 as ℓ increases, whereas the right-most terms tend to β. Thus, a sufficiently large value of ℓ will drive both scores close to their upper bounds, resulting in an overall anomaly score $S(p) = S_w(p)S_t(p) \approx \beta^2$. We recall here that β is the value assigned to words and transitions unseen during training, and the value recommended in KIDS is 2. Consequently, a score of β^2 will inevitably fall beyond any reasonable detection threshold, and hence $\texttt{anom}(p) = \texttt{true}$.

In summary, such a payload p can be used as a probabilistic distinguisher to tell whether d is part of the key or not, since:

- If $d \in D$, then $\texttt{anom}(p) = \texttt{true}$ with probability 1, given a sufficiently large value of ℓ.
- If $d \notin D$, then $\texttt{anom}(p) = \texttt{false}$ with high probability, although dependent on the "quality" of q as discussed above.

4.3.1 Complexity

The existence of false positives in our distinguishing method (i.e., situations when $d \notin D$ but nevertheless $\texttt{anom}(p) = \texttt{true}$) is due to using a q of "poor quality", as explained above. Such false positives can be ruled out by repeating the process with different q's and determining D as the intersection of all the resulting keys. Note that, in doing so, the existence of *just one good payload* in the set suffices to recover the correct key. As a consequence, the complexity of this attack is slightly higher than for the case of the gray-box setting: Again, each trial makes exactly 256 queries to KIDS, and several trials should be attempted to rule out possible false positives. If T is the number of normal payloads available to the attacker, then the attack requires $T \cdot 256$ queries, plus the cost of computing an intersection. A description of the attack is given in Fig. 4.

Since the attack succeeds if there is at least one appropriate q, the overall probability of correctly recovering the key after T attempts is

$$P_{BB}(T) = 1 - \Big(1 - P(q_i)\Big)^T \tag{25}$$

Black-box Key Recovery

Input:
Set of payloads $Q = \{q_i\}_{i=1}^T$ s.t. $\texttt{anom}(q_i) = \texttt{false}$, $|q_i|$ is high and $S(q_i)$ is low
Word w_2 s.t. $n(w_2) = 0$
Parameter $\ell > 1$

Algorithm:

1. **for each** $q_i \in Q$ **do**
2. $\quad D_i \leftarrow \emptyset$
3. $\quad$ **for** $d = 0$ **to** 255 **do**
4. $\quad\quad p \leftarrow (q_i \parallel d \parallel w_2 \parallel d \parallel \cdots \parallel d \parallel w_2)$
5. $\quad\quad$ **if** $\texttt{anom}(p) = \texttt{true}$ **then**
6. $\quad\quad\quad D_i \leftarrow D_i \cup \{d\}$
7. $\quad\quad$ **end-if**
8. $\quad$ **end-for**
9. **end-for**
10. **return** $D = \bigcap_{i=1}^{T} D_i$

Fig. 4. Key-recovery attack on black-box KIDS.

where $P(q_i) = \text{Prob}\big[S(q_i) + \Delta < \tau\big]$. The probability of success increases exponentially with T. For example, if $P(q_i) = 0.5$, using only $T = 4$ payloads yields a probability of success close to 94%, while increasing the number of payloads to $T = 5$ and $T = 6$ gives, respectively, a probability of 97% and 98%. Additionally, in scenarios where the attacker does not know a valid w_2, this can be just guessed as described in the next section. In this case, however, an additional checking must be carried out at the end of the attack in order to find out if w_2 was correct. If the check fails, then the attack has to be re-launched with a different candidate word.

4.4 Obtaining words w_1 and w_2

A crucial assumption in the attacks presented above is that the adversary knows two words, w_1 and w_2, such that $n(w_1) > 0$ and $n(w_2) = 0$. We next describe how such words can be obtained with additional queries to KIDS. Our procedure exploits carefully constructed payloads that certainly are border cases. This forces us to make some assumptions about how KIDS processes such payloads, as they are not covered by the discussion given in the original paper. The first one has been already mentioned: if a payload p has no transitions, then $S_t(p) = \alpha$ and, therefore, $S(p) = S_w(p) \cdot \alpha$. This will be needed when a payload consists of a single word, or a word followed by a delimiter. We also assume that the empty word ε is not a valid word. Thus, if several delimiters appear together in a subsequence of the form $\xi_i \parallel d \parallel d \parallel \cdots \parallel d \parallel \xi_j$, with ξ_i, ξ_j words and $d \in D$, the tokenization process will return only words ξ_i and ξ_j, together with the transition $\xi_i \to \xi_j$. Finally, we assume that words are extracted from a payload through a tokenization process governed by delimiters and not by the words themselves. In other words, this means

Find w_1 (Gray-box Setting)

Input:
Payload p s.t. $\texttt{anom}(p) = \texttt{false}$
Parameters α and β

Algorithm:
1. $q \leftarrow \varepsilon$ *// Empty payload*
2. $i \leftarrow 0$
3. **do**
4. $\quad i \leftarrow i + 1$
5. $\quad q \leftarrow q \parallel p[i]$
6. $\quad s = S(q)$
7. **while** $(s \leq \beta)$ **or** $(s \geq \beta^2)$
8. $w_1 \leftarrow p[1] \parallel \cdots \parallel p[i-2]$
9. $d \leftarrow p[i-1]$
10. **return** w_1 and d

Fig. 5. Algorithm to find w_1 and the first delimiter.

that delimiters are first located, and then every sequence between two delimiters is considered a word. We finally remark that these assumptions are not critical for our attacks, and that variations of the same underlying ideas can be easily derived for other implementations of KIDS.

The procedures described below make use of the same adversarial model where they apply. For the gray-box attack, w_1 and w_2 are obtained by analyzing the anomaly score of a sequence of probing payloads. For the black-box attack, we show how to randomly construct one such w_2 (for w_1 is not needed in this attack) and detect if the choice was correct or not. Furthermore, the algorithms given below assume that α and β are known. We believe that this is reasonable, as these are just tuning parameters and not part of the key. Even if for some reason they are unknown, our attacks could be easily modified to work with estimates (which, incidentally, may be quite accurate given the role that both parameters play in KIDS). Due to space reasons, we do not discuss such modifications here.

4.4.1 Gray-box Setting: Finding w_1

Fig. 5 presents a procedure to recover the first word w of a payload, together with the delimiter d located right after it. The algorithm takes as input a normal payload p (i.e., $\texttt{anom}(p) = \texttt{false}$) that must be of the form: $p = w \parallel d \parallel t$, where:

(i) The first word w is such that $n(w) > 0$.
(ii) d is the first delimiter in p.
(iii) The tail t, possibly composed of several words and delimiters, is such that $n(t[1]) = 0$; i.e., the first byte of t is not a previously seen word.

We suggest to use just a normal p in this algorithm, as in our experience most of them satisfy these conditions.

The procedure shown in Fig. 5 relies on a simple intuition to detect the presence of d: if KIDS is fed with increasingly longer versions of p, the resulting anomaly score will change once the first transition is incorporated into the query. Assume that d is located in the k-th byte of p, i.e., $d = p[k]$ and $w = p[1] \parallel \cdots \parallel p[k-1]$. Let $q^{(i)} = p[1] \parallel \cdots \parallel p[i]$ be the payload used to query KIDS at the i-th iteration. There are two possibilities about the anomaly score of $q^{(i)}$:

(a) $i \leq k$.
In this case, $q^{(i)}$ is always either a prefix of w, or w, or $w \parallel d$. If $q^{(i)}$ is a prefix of w that was not seen during training, then

$$S(q^{(i)}) = \beta\alpha \tag{26}$$

In the $(k-1)$-th and k-th iterations, $q^{(i)}$ is w and $w \parallel d$, respectively. In both cases we have

$$S(q^{(i)}) = \left(\frac{1}{n(w)}\right)\alpha \tag{27}$$

It may occur that a prefix of w is also a word seen during training, so the score given by (27) would be obtained for some $i < k-1$. As it should be clear later, this does not affect the analysis, as the algorithm keeps processing p until $i = k+1$ and always returns w.

(b) $i = k+1$.
In this case, $q^{(i)} = w \parallel d \parallel t[1]$, which yields an anomaly score

$$S(q^{(i)}) = \left[\frac{1}{2}\left(\frac{1}{n(w)} + \beta\right)\right]\beta = \frac{1}{2}\beta^2 + \frac{1}{2n(w)}\beta \tag{28}$$

Note that

$$\frac{1}{2}\beta^2 + \frac{1}{2n(w)}\beta > \beta \quad \text{iff} \quad \beta > 2\left(1 - \frac{1}{2n(w)}\right) \tag{29}$$

which always holds if $\beta \geq 2$, and that

$$\frac{1}{2}\beta^2 + \frac{1}{2n(w)}\beta < \beta^2 \quad \text{iff} \quad \beta > \frac{1}{n(w)} \tag{30}$$

which is also true. Consequently, for case (b) we have that $\beta < S(q^{(i)}) < \beta^2$.

The correctness of the algorithm is based on the fact that the score given by (28), which is always in the open interval (β, β^2), is different from (26) and from (27) with high probability. However, the particular value chosen for parameter α is relevant here. As discussed above, it is reasonable to assume that either $\alpha = 1$ (so transitions do not count in payloads without them) or $\alpha = \beta$ (having no transitions is considered an anomaly equal to any other). In either case, both (26) and (27) fall out of (β, β^2) and the algorithm succeeds with probability 1. Thus, the transition triggered by delimiter d can be recognized by just checking that the anomaly score falls within (β, β^2).

There is one scenario where the algorithm will fail: if α is exactly $\frac{1}{2}\left(\frac{1}{n(w)} + \beta\right)$, i.e., for the specific w contained in this payload p, then (26) and (28) coincide. In this case, the algorithm's loop (step 7 in Fig. 5) does not stop after the first delimiter, but after the first word whose count differs from that involved in the α value. The overall result is that the word w returned by the algorithm would actually be composed of two or more

words separated by delimiters. Recall, however, that this particular choice for α prevents the gray-box attack from succeeding (see discussion at the end of Section 4.2), so the fact that w is incorrect has little importance. Recall, too, that this is unlikely to happen as the defender will not know which particular w the attacker would be using and, besides, it has to be done only for just one particular w.

Complexity. The number of queries to KIDS required to find one valid w_1 depends on two factors. On the one hand, each run of the algorithm makes exactly $|w_1|+2$ queries (we do not consider here the case when the algorithm fails because of the choice of α discussed above). The average word length is, in turn, related to the key size, with words becoming generally shorter when the key consists of more delimiters, although this also depends on the underlying generative model for payloads (i.e., the probability of observing each byte at each position). In general, it is expected that $|w_1|$ will be low for payloads associated with usual network traffic (e.g. HTTP or FTP services). On the other hand, the algorithm fails if p does not satisfy the requirements assumed above, i.e., having a first word with positive count and the first byte of the second with zero count. Let p_{w_1} be the probability of a payload satisfying these conditions. In our experience, this occurs extremely often, since the first portion of the payload generally transports protocol signaling (e.g., service/resource names) very common among payloads, making p_{w_1} close to 1.

In summary, each run of the algorithm can be seen as an experiment that makes $|w_1|+2$ queries and succeeds with probability p_{w_1}. Thus, the probability of generating at least one valid w_1 after n trials of the algorithm (assuming independent trials) is

$$P_1(n) = 1 - (1 - p_{w_1})^n \tag{31}$$

Note that the probability of success increments exponentially with n, but only requires a linear ($|w_1|+2$) number of additional queries to KIDS. For example, assuming that $p_{w_1} = 0.9$, the probability of finding w_1 with two $n = 2$ runs of the algorithm (i.e., $2|w_1|+4$ queries) is equal to 0.99, whereas for $p_{w_1} = 0.75$ it takes $n = 3$ runs ($3|w_1|+6$ queries) to reach a similar probability.

4.4.2 Gray-box Setting: Finding w_2

In the gray-box setting, finding w_2 such that $n(w_2) = 0$ is straightforward given that w_1 and a delimiter d have been previously obtained. Assume that $b \neq d$ is a randomly chosen byte and that w_2 consists of λ repetitions of b. Consider now the payload $q = w_2 \parallel d \parallel w_1$:

(i) If $n(w_2) = 0$, then

$$S(q) = \left[\frac{1}{2}\left(\beta + \frac{1}{n(w_1)}\right)\right]\beta \tag{32}$$

with $S(q) > \beta$, as discussed above in (29).

(ii) If $n(w_2) > 0$, then

$$S(q) = \left[\frac{1}{2}\left(\frac{1}{n(w_2)} + \frac{1}{n(w_1)}\right)\right]\frac{1}{n(w_2 \rightarrow w_1)} \tag{33}$$

with $S(q) < \beta$.

(ii) Finally, if $b \in D$, then $S(q) = S(w_1) < \beta$.

Thus, a suitable w_2 can be found by trying different values of b while the anomaly score of q falls below β. The overall procedure is summarized in Fig. 6.

Find w_2 (Gray-box Setting)

Input:
Payload p s.t. $\texttt{anom}(p) = \texttt{false}$
Parameters α, β, and m
Algorithm:
1. $(w_1, d) \leftarrow \text{find\_}w_1(p, \alpha, \beta)$
2. **do**
3. $b \; \epsilon_R \; [0, 255]$
4. $w_2 \leftarrow \overbrace{b \parallel \cdots \parallel b}^{\lambda}$
5. $q \leftarrow w_2 \parallel d \parallel w_1$
6. $s = S(q)$
7. **while** $(s < \beta)$
8. **return** w_2

Fig. 6. Algorithm to find w_2.

Complexity. The number of queries to KIDS required to find w_2 is related to the probability of randomly choosing one $b \notin D$ and the probability p_{w_2} of the resulting w_2 being a word unseen during training. If we assume that key elements are drawn randomly and independently, then the probability of generating at least one valid w_2 after m queries to KIDS is

$$P_2(m) = 1 - \left[1 - \left(\frac{256 - |D|}{256}\right)p_{w_2}\right]^m \tag{34}$$

which, again, increments exponentially with m. For key sizes ranging from 15 to 30, as suggested in [12], the probability of finding a valid w_2 with just 1 query falls between $0.92 \cdot p_{w_2}$ and $0.94 \cdot p_{w_2}$. Using some domain-specific knowledge may help the attacker in selecting candidate words so that p_{w_2} is close to 1. For the purposes of this paper, we have chosen words consisting of λ repetitions of the same byte b. This has proven to be adequate for payloads of most application protocols, as such strings are not generally found in them, particularly for values of λ sufficiently high (say, $\lambda > 10$). Thus, assuming that $p_{w_2} \approx 1$, the probability of finding w_2 with just one query to KIDS is greater than 0.9. Using similar reasoning, it is straightforward to see that $P_2(2) > 0.99$.

4.4.3 Black-box Setting: Finding w_2

The procedure given above to find w_2 cannot be applied to the black-box setting since it requires access to the anomaly score of the probing payloads. In this case, we suggest to use the attack described in Section 4.3 with a word w_2 generated as in the previous algorithm, i.e., composed of a randomly chosen byte b repeated λ times. The analysis is then equivalent to the one given in the complexity analysis of Section 4.4.2: If (i) $b \notin D$; and (ii) the resulting w_2 is a word unseen during training,

TABLE 1
Summary of the attacks presented in this paper with their complexity and the probability of success.

Setting	Procedure	Number of queries to KIDS	Probability of success
Gray box	Finding w_1	$n \cdot (\|w_1\| + 2)$	$P_1(n) = 1 - (1 - p_{w_1})^n$
	Finding w_2	m (assuming w_1 is known)	$P_2(m) = 1 - \left[1 - \left(\frac{256 - \|D\|}{256}\right) p_{w_2}\right]^m$
	Key recovery	257	1
	Total	$n \cdot (\|w_1\| + 2) + m + 257$	$P_1(n) \cdot P_2(m)$
Black box	Finding w_2	0	$P_2(m)$
	Key recovery	$T \cdot 256$	$P_{BB}(T) = 1 - \left(1 - P(q_i)\right)^T$
	Total	$m \cdot T \cdot 256$	$P_2(m) \cdot P_{BB}(T)$

then the attack succeeds with probability $P_2(m) \cdot P_{BB}(T)$. However, if the chosen byte b turns out to be a delimiter (i.e., $b \in D$, with occurs with probability $\frac{|D|}{256}$), the algorithm behaves differently. Recall that the attack iterates over all possible delimiters d (line 3 in Fig. 4). Now:

- **Case 1:** $d \in D$
  In this case, the tail $t = d \parallel w_2 \parallel d \parallel w_2 \parallel d \parallel \cdots \parallel d \parallel w_2$ used in the payload $p = q \parallel t$ will be discarded by the tokenization process as it contains no words. As $S(p) = S(q)$, the payload will inevitably be labeled as normal and the attack will not take the delimiter d as belonging to the key. The overall result is that all true delimiters $d \in D$ will not be identified. This happens with probability 1.
- **Case 2:** $d \notin D$
  In this case, the tail t is split into ℓ words. Note, however, that now w_2 (in particular, just one b) plays the role of the delimiter and d is considered a word. The result depends on whether $n(d)$ is strictly greater than zero or not:
  (a) $n(d) = 0$
  This situation is equivalent to case 2 in Section 4.3: p is split into $k + \ell$ words, with a tail full of previously unseen words and transitions. In this case, $\texttt{anom}(p) = \texttt{true}$ and the algorithm incorrectly takes d as a true delimiter.
  (b) $n(d) > 0$
  In this case, the result depends on whether transitions of the form $d \to d$ have or have not a positive count. If $n(d \to d) = 0$, then the transition score S_t of the tail will be very high, possibly making the overall score of p anomalous. The details are similar to those discussed for case 2 in Section 4.3, although only applying to $S_t(p)$. As in the case above, the result is that d is incorrectly taken as a true delimiter. However, if $n(d \to d) > 0$, the result in unpredictable as it depends on the final score of p. Therefore, d might be discarded or not.

In summary, we have:

- If $b \notin D$ and the chosen word w_2 is good, the attack succeeds with the aforementioned probability.
- If $b \in D$, the roles of words and delimiters are swapped. In this case, the output of the algorithm is a subset of D^C (the complement of D). In particular, it consists of at least all individual bytes $e \notin D$ such that $n(e) = 0$ or $n(e \to e) = 0$.

The second case occurs with probability $\frac{|D|}{256}$. As key sizes suggested in [12] are relatively small (from 15 to 30), this knowledge could be used to tell whether the returned key is the true key or not by simply inspecting its size. Furthermore, even if the obtained D is too large to be the true key, knowing that it is a subset of D^C is still quite valuable, as the true key can be estimated by just taking the complement. A corollary of this result is that the size of D should be kept secret too. However, in the case of KIDS there may be some fundamental limitation, as keys composed of too few or too many delimiters might not produce useful detection models. This needs to be further investigated.

4.4.4 Attack Complexity Revisited

To conclude this section, in Table 1 we summarize the overall complexity of the attacks presented in this paper, including the procedures used to find w_1 and w_2 discussed above.

4.5 Experimental Results

We have experimentally validated our attacks with an implementation of KIDS written in C. The system was trained with 2000 HTTP payloads captured in a university network. The dataset does not include attacks, as they are not necessary to recover the key. Following the design principles given in [12], our experiments have been conducted with key sizes ranging from 15 to 30, even though this parameter has little influence on the results. In all cases, the delimiters are randomly generated avoiding repetitions, and the detection threshold is chosen to guarantee that at least 99% of the training set falls below it. We note that this way of selecting a key does not coincide with the procedure given in [12], where the authors suggest a method involving both normal and attack traffic. This, however, is irrelevant to our attacks, as they work on an already trained system regardless of how the key has been chosen.

In the case of the gray-box attacks, words w_1 and w_2 are automatically extracted from one normal payload by

following the procedures described in Section 4.4. According to our practical experience, the values of n, $|w_1|$, and m (see Table 1) remain very low for the scenarios tested. For example, w_1 is consistently recovered from just one payload (i.e., $n = 1$ and the algorithm never fails) and these words rarely have more than $|w_1| = 15$ bytes. This conforms to the intuition given above, but it is reinforced by the fact that we used HTTP traffic where the first bytes of every payload refer to websites and resources already present during training. As for w_2, we used $\lambda = 16$ and systematically obtained a valid w_2 in less than $m = 3$ attempts. We ran the attack multiple times with randomly generated keys and, in all cases, we correctly recovere all the delimiters as expected.

For the black-box attacks, we used a subset of T randomly chosen payloads and made them available to the attacker. Different combinations of T and the paramenter ℓ were tried. As anticipated, the probability of correctly recovering all the key elements increases both with T and, especially, with ℓ. In fact, a low value of T suffices if it contains "good" payloads, as defined in Section 4.3, whereas the success probability dramatically decreases for low values of ℓ. In our case, values $T \approx 5$ and $\ell > 10$ proved enough to correctly recover the key. Nevertheless, we emphasize that such parameters will generally be very dependent on the specific dataset used to train the system.

In practice, an important issue is the overall time required by the attacker to recover the key, particularly when access to the system is given for a short period of time. In general, such time is:

$$t_{attack} = N_q(t_{pre} + t_s + t_q + t_r + t_{post}) + t_{loc} \qquad (35)$$

where

- N_q is the number of queries to KIDS involved in the attack.
- t_{pre} is the time required to prepare the query.
- t_s is the time taken by the payload to reach KIDS.
- t_q is the time taken by KIDS to process the payload.
- t_r is the time taken for the result to reach the attacker.
- t_{post} is the time required to process the result.
- t_{loc} is the time taken by the local computations made by the key-recovery algorithm

In our experiments, t_{pre}, t_{post} and t_{loc} are negligible. The average time to process a payload (t_q) is given in Table 2. Even though this time actually depends on the payload length, we have found that the variation is negligible for the values involved in our attacks. Finally, both t_s and t_r strongly depend on the attack setting. Thus, if the attack is carried out remotely, their sum will roughly be equal to the average network latency, measured by the Round Trip Time (RTT). Finally, the number of queries depends on the specific attack, with $N_q = 257$ for the gray-box model and $N_q = T * 256$ for the black-box model.

Tables 3 and 4 show the total key-recovery time in two different settings. In the first one, the RTT between the attacker and KIDS was around 50 ms (low latency), whereas in the second it was around 195 ms (high latency). In both cases, the results were averaged over 50 executions, each one with a different randomly chosen key. For the gray-box attacks, recovering the key takes around 13 s for the low-latency scenario and around 50 s for the high-latency network. Each one of the T iterations of the black-box attack takes a similar time, so the key can be recovered in approximately one minute for a low-latency network, and in around 5 minutes for the high-latency setting. As a final note, It must be emphasized that, in practice, the dominant factor in the attack efficiency is the sum $t_s + t_r$, i.e., the time taken to send each query and get the result.

TABLE 2
Average time taken by our KIDS implementation to process a payload

Key size	15	20	25	30
Time (ms)	0.093 ms	0.169 ms	0.289 ms	0.382 ms

TABLE 3
Experimental average time (in seconds) to recover the key by the gray-box attack

Key size	Low latency network	High latency network
15	11.27 s	49.15 s
20	12.98 s	49.62 s
25	13.03 s	50.06 s
30	13.29 s	50.73 s

TABLE 4
Experimental average time (in seconds) to recover the key by the black-box attack

T	Key size	Low latency	High latency
4	15	50.78 s	198.11 s
	20	50.98 s	199.86 s
	25	51.26 s	199.92 s
	30	52.03 s	201.31 s
5	15	62.12 s	245.12 s
	20	62.47 s	246.48 s
	25	62.55 s	247.78 s
	30	63.98 s	248.62 s
6	15	75.46 s	291.31 s
	20	76.08 s	291.68 s
	25	76.84 s	293.83 s
	30	77.93 s	294.06 s

5 Discussion: Keyed Anomaly Detection and Adversarial Models Revisited

We conclude this paper by revisiting the idea of keyed anomaly detection (or, more generally, keyed classification) and further discussing what realistic adversarial models should be used to assess their security.

Perhaps the first obvious question is whether it makes sense at all to introduce some secret material into a

learning algorithm so as to make evasion harder. To the best of our knowledge, all the approaches explored so far to counteract evasion fall into one of the two strategies discussed in Section 2, namely randomizing the classification process (e.g., [3], [20]) or optimally adapting it from a cost-sensitive perspective (e.g., [5]). Anagram [22] is a special case, since it explicitly possesses the notion of a "key" (bitmasks used to choose what parts of the payload will be analyzed). Unfortunately, we are not aware of any work studying the strength of Anagram against key-recovery attacks.

All in all, we believe that the idea of learning a classifier with a key is worth exploring. However, we suggest two fundamental properties that any such keyed scheme must explicitly address:

1) The designers must prove, or at least give sound heuristic arguments, that evasion is computationally infeasible[1] without knowing the key.
2) It must be proved that recovering the key is computationally infeasible under reasonable adversarial models, e.g., it cannot be done with polynomially-many queries.

Some recent works have raised similar questions when discussing models and challenges for the classifier evasion problem. For instance, Nelson et al. explictly mention in [13] that it may be interesting to consider adversarial models beyond the membership oracle. Thus, if a classifier is defined as $f(\mathbf{x}) = \mathbb{I}\{g(\mathbf{x}) > 0\}$ for some function g, what if the attacker receives $g(\mathbf{x})$ for every query rather than just $f(\mathbf{x})$, i.e., the '+'/'-' label? Note that this is precisely the case we have explored under our gray-box model. In our case, such information has proven to be essential to reduce the attack complexity.

Closely related to the points discussed above is the need to establish clearly defined and motivated adversarial models for secure machine learning algorithms. The assumptions made about the attacker's capabilities are critical to properly analyze the security of any scheme, but some of them may well be unrealistic for many applications. One debatable issue is whether the attacker can really get feedback from the system for instances he chooses. This bears some analogies with Chosen-Plaintext Attacks (CPA) in cryptography. This assumption has been made by many works in secure machine learning, including ours. In our opinion, it would be unsafe to assume that the attacker does not have such an ability, even if we cannot figure out how he would do in practice. Furthermore, this is not incompatible with analyzing the security of a scheme against a weaker model where interactions are more restricted.

According to the attack model introduced by Barreno et al. [2], the class of attacks discussed here are *exploratory*, as they attempt to discover information once the classifier has been learned. However, it seems worth exploring the resilience of a keyed classifier against *causative* attacks, i.e., scenarios where the adversary strategically participates in the training process by providing carefully constructed samples. Some recent works have started to look into this matter (e.g., [4]). However, it is unclear to us what protection against such attacks a keyed classifier might provide. For example, does the fact that a secret key is used prevent the attacker from forcing the training process into learning an undesirable concept (e.g., one that includes attacks)? A priori, this seems unlikely. If that is the case, then the very notion of a keyed classifier will provide protection against evasion attacks only, and assuming that the attacker has no control whatsoever over the training process.

Finally, given that KIDS does not meet the security requirements discussed above, one natural question is: Where to put the key then? The intuition dictates that a keyed classifier must learn a *key-dependent* and *secret* concept, meaning that an adversary must not be able to guess it (entirely or approximately) without knowing the key. But, simultaneously, the classifier must classify well, which introduces one apparently fundamental limitation: If the adversary has access to the training data distribution, nothing stops him from building his own classifier, keyed or not, which will necessarily be a fairly good approximation (in terms of the classification boundary) of the one to be attacked. Consequently, in a keyed classifier the focus may not be on hiding the classification boundary, but on introducing, from an attacker's perspective, *sufficient* uncertainty about how samples are processed. This is the core idea behind the use of randomized classifiers. The challenge is whether the same can be done in a key-dependent way.

6 Conclusions

In this paper we have analyzed the strength of KIDS against key-recovery attacks. In doing so, we have adapted to the anomaly detection context an adversarial model borrowed from the related field of adversarial learning. We have presented key-recovery attacks according to two adversarial settings, depending on the feedback given by KIDS to probing queries.

To the best of our knowledge, our work is the first to demonstrate key-recovery attacks on a keyed classifier. Surprisingly, our attacks are extremely efficient, showing that it is reasonably easy for an attacker to recover the key in any of the two settings discussed. We believe that such a lack of security reveals that schemes like KIDS were simply not designed to prevent key-recovery attacks. However, in this paper we have argued that resistance against such attacks is essential to any classifier that attempts to impede evasion by relying on a secret piece of information. We have provided discussion on this and other open questions in the hope of stimulating further research in this area.

The attacks here presented could be prevented by introducing a number of ad hoc countermeasures to

1. We adopt here the standard notion of computational security common in many branches of cryptography. Informally speaking, this means that breaking the system reduces to solving a hard problem.

the system, such as limiting the maximum length of words and payloads, or including such quantities as classification features. We suspect, however, that these variants may still be vulnerable to other attacks. Thus, our recommendation for future designs is to base decisions on robust principles rather than particular fixes.

Going beyond KIDS, it remains to be seen whether similar schemes (e.g., Anagram [22]) are secure against key-recovery attacks. As discussed in Section 1, our attacks (or variants of them) are focused on keyed classifiers, and we believe that they will not carry over randomized classifiers. We note that, in its present form, KIDS cannot be easily randomized, as choosing a new key implies training the classifier again, which is clearly impractical in real-world scenarios. In the case of Anagram, the authors discuss one mode of operation where one key (a secret but fixed bitmask) is used to split the packet in various portions so that each of them is checked against a different Bloom filter. This scheme bears numerous resemblances to KIDS and the key may be recovered with attacks similar to those presented in this paper. Nevertheless, this needs further investigation and will be addressed in future work.

Our focus in this work has been on recovering the key through efficient procedures, demonstrating that the classification process leaks information about it that can be leveraged by an attacker. However, the ultimate goal is to evade the system, and we have just assumed that knowing the key is essential to craft an attack that evades detection or, at least, that significantly facilitates the process. It remains to be seen whether a keyed classifier such as KIDS can be just evaded without explicitly recovering the key. If the answer is in the affirmative, then the key does not ensure resistance against evasion.

REFERENCES

[1] M. Barreno, B. Nelson, R. Sears, A.D. Joseph, and J.D. Tygar. "Can Machine Learning Be Secure?" In *ASIACCS 2006*, pp. 16–25, 2006.

[2] M. Barreno, B. Nelson, A.D. Joseph, and J.D. Tygar. "The security of machine learning." In *Machine Learning*, 81(2):121–148, 2010.

[3] B. Biggio, G. Fumera, and F. Roli. "Adversarial Pattern Classification Using Multiple Classifiers and Randomisation." In *Proc. 2008 IAPR Intl. Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pp. 500–509. Springer-Verlag, 2008.

[4] B. Biggio, B. Nelson, P. Laskov. "Support Vector Machines Under Adversarial Label Noise." In *Journal of Machine Learning Research - Proceedings Track*, Vol. 20, pp. 97–112, 2011.

[5] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. "Adversarial classification." In *KDD 2004*, pp. 99–108, 2004.

[6] P. Fogla, M. Sharif, R. Perdisci, O. Kolesnikov, and W. Lee. "Polymorphic blending attacks." In *USENIX Security Symp.*, 2006.

[7] C. Gates and C. Taylo. "Challenging the anomaly detection paradigm: A provocative discussion." In *New Security Paradigms Workshop (NSPW)*, pp. 21–29, 2006

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

[9] O. Kolesnikov, D. Dagon, and W. Lee. "Advanced polymorphic worms: Evading IDS by blending in with normal traffic." In *USENIX Security Symposium*, 2005.

[10] D. Lowd and C. Meek. "Adversarial learning." In *KDD 2005*, pp. 641–647, 2005.

[11] Metasploit Framework. Available at: www.metasploit.com.

[12] S. Mrdovic and B. Drazenovic. "KIDS - Keyed Intrusion Detection System." In *DIMVA 2010*, LNCS 6201, pp. 173–182, 2010.

[13] B. Nelson, B.I.P. Rubinstein, L. Huang, A.D. Joseph, and J.D. Tygar. "Classifier evasion: models and open problems." In *PSDML 2010*, pp. 92–98. Springer-Verlag, 2011.

[14] B. Nelson, A.D. Joseph, S.J. Lee, and S. Rao. "Near-Optimal Evasion of Convex-Inducing Classifiers." In *Journal of Machine Learning Research - Proceedings Track*, Vol. 9, pp. 549–556, 2010.

[15] B. Nelson, B.I.P. Rubinstein, L. Huang, A.D. Joseph, S.J. Lee, S. Rao, and J.D. Tygar. "Query Strategies for Evading Convex-Inducing Classifiers." In *Journal of Machine Learning Research*, Vol. 13 (May), pp. 1293–1332, MIT Press, 2012

[16] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee. "McPAD: A multiple classifier system for accurate payload-based anomaly detection." In *Computer Networks*, Vol. 5, No. 6, pp. 864–881, 2009

[17] K. Rieck. "Computer Security and Machine Learning: Worst Enemies or Best Friends?" In *DIMVA Workshop on Systems Security (SYSSEC)*, 2011.

[18] R. Sommer and V. Paxson. "Outside the closed world: On using machine learning for network intrusion detection." In *IEEE Symposium on Security and Privacy*, pp. 305–316, 2010.

[19] Y. Song, M. Locasto, A. Stavrou, A.D. Keromytis, and S.J. Stolfo. "On the infeasibility of modeling polymorphic shellcode: Rethinking the role of learning in intrusion detection systems." In *Machine Learning*, Vol. 81, No. 2, pp. 179–205, 2010.

[20] J.E. Tapiador and J.A. Clark. "Masquerade mimicry attack detection: A randomised approach." In *Computers & Security* 30(5):297–310. Elsevier Ltd., 2011.

[21] K. Wang, G. Cretu, and S. Stolfo. "Anomalous Payload-Based Worm Detection and Signature Generation." In *RAID 2005*, pp. 227–246. Springer-Verlag, 2005.

[22] K. Wang, J. Parekh, and S. Stolfo. "Anagram: A content anomaly detector resistant to mimicry attack." In *RAID 2006*, pp. 226–248. Springer-Verlag, 2006.

[23] Y. Zhou, Z. Jorgensen, and M. Inge. "Combating good word attacks on statistical spam filters with multiple instance learning." In *ICTAI 2007*, pp. 298–305, 2007.

Juan E. Tapiador is Associate Professor of Computer Science at Universidad Carlos III de Madrid, Spain. Prior to joining UC3M, he was Research Associate at the University of York, UK. His main research interests are in applied cryptography and network security. He holds a M.Sc. in Computer Science from the University of Granada (2000), where he obtained the Best Student Academic Award, and a Ph.D. in Computer Science (2004) from the same university.

Agustin Orfila is Associate Professor in the Computer Science Department at Universidad Carlos III de Madrid. He has a B.Sc. in Physics from Universidad Complutense de Madrid and a Ph.D. in Computer Science from Universidad Carlos III de Madrid. His main interests lie in the field of network and computer security, particularly in Intrusion Detection Systems and RFID Systems.

Arturo Ribagorda is Professor at Universidad Carlos III de Madrid, where he also serves as Head of the Computer Security Lab in the Computer Science Department. He has a M.Sc. in Telecommunications Engineering and a Ph.D. in Computer Science. He has more than 25 years of R&D experience in computer and information security, and has authored 4 books and more than 100 articles in these areas. He also serves as program committee member for several conferences related to cryptography and information security.

Benjamin Ramos is Associate Professor in the Computer Science Department at Universidad Carlos III de Madrid. He has a B.Sc. in Mathematics and a Ph.D. in Computer Science from Universidad Carlos III de Madrid. His main interests lie in the field of computer security.