A Learning-based Approach to Secure JTAG against Unseen Scan-based Attacks

Xuanle Ren[1,2], R. D. (Shawn) Blanton[2], and Vítor Grade Tavares[1]

[1]*INESC TEC and Faculty of Engineering, University of Porto, Porto, Portugal*
[2]*Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA*

***Abstract*—Security is becoming an essential problem for integrated circuits (ICs). Various attacks, such as reverse engineering and dumping on-chip data, have been reported to undermine IC security. IEEE 1149.1, also known as JTAG, is primarily used for IC manufacturing test but inevitably provides a "backdoor" that can be exploited to attack ICs. Encryption has been used extensively as an effective mean to protect ICs through authentication, but a few weaknesses subsist, such as key leakage. Signature-based techniques ensure security using a database that includes known attacks, but fail to detect attacks that are not contained by the database. To overcome these drawbacks, a two-layer learning-based protection scheme is proposed. Specifically, the scheme monitors the execution of JTAG instructions and uses support vector machines (SVM) to identify abnormal instruction sequences. The use of machine learning enables the detection of unseen attacks without the need for key-based authentication. The experiments based on the OpenSPARC T2 platform demonstrate that the proposed scheme improves the accuracy of detecting unseen attacks by 50% on average when compared to previous work.**

I. Introduction

Security is becoming a central problem for integrated circuits (ICs). Among the potential attacks, scan-based attacks are performed through the standard test access port and the boundary-scan architecture which is defined by the IEEE 1149.1 (or named JTAG). Scan-based attacks are one of the most commonly exploited methods due to two main reasons [1]–[3]. First, the JTAG is widely used as an interface for manufacturing testing and in-field debugging in modern ICs [4]. So it is easy to find the test access ports in most chips. Second, the JTAG provides powerful features to access on-chip data, such as firmware and on-chip memory. For example, the OpenSPARC T2 benchmark [5] has more than ten debugging functions accessible through the JTAG, e.g., L2 cache r/w, memory built-in self-test (MBIST), shadow scan, direct memory observation, etc. All these functions are undocumented, that is, unauthorized users do not know which JTAG functions are implemented and how they are operated. In [6], an attacking flow is described for reverse engineering undocumented JTAG functions within the OpenSPARC T2.

The attackers may have different levels of access to the IC and thus may employ different types of attacks, e.g., analyzing power consumption and/or magnetic emanation, influencing IC fabrication process, modifying the communication within the IC, etc. In this paper, it is assumed that the attacker can only access the JTAG ports. Although some attackers with greater levels of access are able to utilize more powerful tools, it is still reasonable to assume that the JTAG port is an essential conduit for accessing the IC [1], [2].

Various approaches have been proposed to protect the JTAG from being misused, including disabling the JTAG before shipping the IC to customers [7], obfuscating the JTAG outputs [8], etc. The disadvantages however include hampering of in-field debugging and/or limiting JTAG use to public functions. Encrypting the JTAG using plain-text passwords or more complex protocols is deemed a more secure technique since user authentication is required [9]–[15]. However, it modifies IEEE 1149.1 due to the support of lock/unlock commands, and suffers from the problem of password leakage.

To protect the private JTAG functions and also to overcome the drawbacks of the encryption techniques, detecting attacks by monitoring user behavior is becoming a potential approach [6], [16]–[18]. One example is signature detection, which attempts to identify events that misuse the system by creating models of attacks (called signatures), and therefore knowledge of normal events is not required for these approaches [16]. Observed behavior that matches any of the signature is labeled as an attack. However, the effectiveness of a signature detector strongly relies on the completeness of the attack models. Anomaly detection, on the other hand, creates models of normal uses and detects operations that do not conform. An anomaly detector can be achieved by either finite state machines [17] or probabilistic models (e.g., Markov chain [18]). A finite state machine labels all deviations as attacks, while a probabilistic model labels the user behavior with a probability rather than a yes-no decision. However, for both models, variances of normal operation lead to a high false positive rate. The third approach is learning-based models that detect attacks using a classifier. In [6], a decision tree is learned from normal operation and various attack strategies. However, JTAG security is still threatened by novel or unseen attacks that either target a different IC component or exploits a new strategy never encountered before by the classifier.

In this paper, a two-layer learning-based protection scheme is proposed. The scheme mitigates the problem of unseen scan-based attacks and allows variances of normal operation. Layer-I applies a basic check that verifies if basic rules for JTAG operations are violated. Layer-II labels the user behavior (i.e., the sequence of opcodes) as normal or attack using a support vector machine (SVM) classifier. The user behavior

978-1-4673-9039-2/16 $31.00 © 2016 IEEE
DOI 10.1109/ISVLSI.2016.107

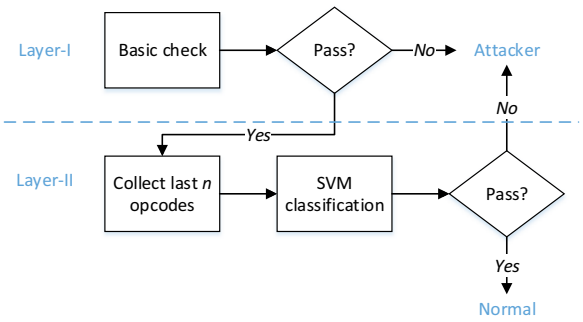


Figure 1: The protection scheme comprises two layers: basic check and SVM classification.

of operating the JTAG is initially examined by layer-I, and then examined by layer-II only if it passes layer-I. Experiments based on the OpenSPARC T2 demonstrate a detection improvement for unseen scan-based attacks by 50% on average when compared to previous work. However, the capability to detect new attacks is difficult to verify comprehensively since different types of attacks are always arising. In the experiments, a known attack is treated as unseen by excluding it from classifier training. The experiments therefore provide a preliminary evaluation of the capability to detect new scan-based attacks.

The rest of this paper is organized as follows. Section II describes the two-layer protection scheme. In Section III, the hardware architecture of the protection scheme is presented and its overhead is estimated. Section IV compares the performance of the scheme with previous approaches. Section V discusses the limitations of the two-layer protection scheme. Finally, Section VI concludes the paper.

II. Two-Layer Protection Scheme

As assumed, the attacker can supply inputs and observe the corresponding outputs via the JTAG ports. In addition, the attacker may apply attacks that are not known by the protection scheme. The proposed approach employs a hierarchical scheme that comprises two layers of protection (Figure 1). The user behavior is initially examined by layer-I, and then examined by layer-II only if it passes layer-I.

A. Layer-I: Basic Check

Basic check is intended to detect attackers that violate the basic rules of JTAG operation. An attacker that knows little to nothing about the IC may operate the JTAG in a manner that substantially differs from normal users. The basic check can block the obvious attacks, including the use of an illegal opcode and/or incorrect length of read/write data. Illegal opcode refers to the bits loaded into the instruction register (IR) that do not correspond to any valid JTAG function. An n-bit IR means that at most 2^n opcodes are available. However, usually only a subset of all the possible opcodes correspond to valid JTAG functions, while others remain unused and are thus considered illegal. The illegal opcodes cannot control or observe any internal signal of the IC, and therefore will not be used by an authorized user. The second misuse involves invoking an incorrect length of read/write data. Specifically, most legal opcodes utilize a data register (DR) in order to perform their functions, and an authorized user should read/write the same length of data as required by the DR. For example, to access on-chip memory, a JTAG opcode for loading the memory address is necessary. However, if an attacker is not aware of the address length, then the length of the supplied input might differ from the correct one.

Both of these misuses can be easily avoided by an authorized user, but not for a user with no prior knowledge. If either misuse is detected, then the user is labeled as an attacker immediately; otherwise, layer-II is invoked.

B. Layer-II: SVM Classification

1) Opcode Sequence: The misuses examined by layer-I might be avoided by an attacker with prior knowledge of the specific JTAG. In layer-II, the sequential order of JTAG instructions is considered because it better characterizes the behavior of a user. Specifically, a sequence of n instructions (i.e., the opcode sequence) is a behavioral pattern of the user utilized for identifying an attacker.

For the OpenSPARC T2 [5], an individual JTAG operation is usually achieved by a sequence of specific instructions. For example, the memory built-in-self-test (MBIST) operation is performed by an instruction sequence of MBIST_BYPASS, MBIST_MODE, MBIST_START and MBIST_RESULT. Another observation is that the opcode sequence length varies for different operations, which makes it a challenging task to identify the starting point and the ending point of a real-time operation. In this work, the opcode sequence length, denoted by n, is set to a fixed value, and the value chosen for n is determined empirically from experiments.

Finally, the opcode sequences of the OpenSPARC T2 are extracted from both normal operation and attacks in an overlapping manner using a fixed-size sliding window.

2) SVM Classification: The extracted opcode sequences (n=4) are visualized in a two-dimensional space using principle component analysis (PCA) as shown in Figure 2 [19]. Specifically, the opcode sequences are "rescaled" in a space that is based on a set of linearly uncorrelated variables (called principle components). The x-axis and y-axis represent the "scale" in terms of the first and the second principle components, respectively. Figure 2 shows that normal operation and attacks not only have a non-linear boundary, but also have overlapping regions. To learn the non-linear boundary, a support vector machine (SVM) is used.

A support vector machine (SVM) is a supervised learning model used for classification. In classification, given a set of samples, each belonging to one out of two classes, a classifier based on these samples (called the training process) is constructed and later used to predict which class a new sample belongs to. In this work, a set of opcode sequences, labeled either normal or attack, are used to train an SVM classifier. In real time, each opcode and the previous $(n-1)$

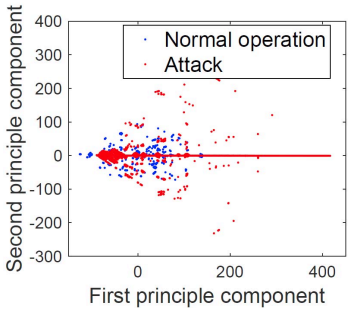
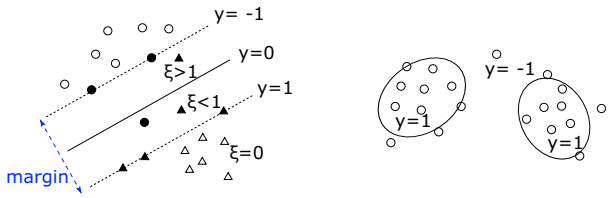


Figure 2: The opcode sequences (n=4) extracted from the OpenSPARC T2 are visualized using PCA.



(a) (b)

Figure 3: (a) An SVM model separates two classes of samples by finding a decision boundary and maximizing the margin. (b) A one-class SVM is trained based solely on normal samples and finds regions where samples form high-density clusters.

opcodes compose a sequence which is then categorized by the classifier.

The objective of an SVM is to find a decision boundary that can separate two classes of samples such that the smallest distance between the decision boundary and any of the samples is maximized (Figure 3(a)). The decision boundary is modeled as

$$y(x) = w^T \phi(x) + b \tag{1}$$

where x is a sample, $y(x)$ is the prediction of x, and $\phi(x)$ is a fixed feature-space transformation. The decision boundary (i.e., the value of w and b) is found by solving

$$\underset{w,\xi}{\operatorname{argmin}} \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n \tag{2}$$

$$s.t. \qquad \|t_n\left(w^T\phi(x_n)\right) + b\| \geq 1 - \xi_n, \quad n = 1, ..., N$$

where x_n is the n-th training sample, t_n is the label (either 1 or -1) of x_n, ξ_n is a slack variable that allows samples to be misclassified with a penalty, and $C > 0$ controls the trade-off between the slack variable penalty and the margin. This optimization problem is convex and to find the solution, its dual representation is derived. More details can be found in [20]. The final solution is

$$y(x) = \operatorname{sgn}\left(\sum_{n\in S} a_n t_n k(x, x_n) + b\right) \tag{3}$$

and

$$b = \frac{1}{N_S}\sum_{n\in S}\left(t_n - \sum_{m\in S} a_m t_m k(x_m, x_n)\right) \tag{4}$$

where S denotes the set of support vectors, N_S represents the number of support vectors, and $k(x, x\prime) = \phi(x)\phi(x\prime)$ denotes the kernel function in terms of x and $x\prime$. The support vectors contain the samples that are located within the margin and the samples that are misclassified. Once the SVM classifier is trained, all the samples are discarded except for those corresponding to the support vectors. The kernel function can be linear or non-linear, which corresponds to a linear boundary or a non-linear boundary, respectively. In this work, the radial basis function (RBF), shown in formula (5), is used as the kernel function.

$$k(x, x\prime) = \exp\left(-\frac{\|x - x\prime\|^2}{2\sigma^2}\right) \tag{5}$$

A one-class SVM is an unsupervised learning algorithm used for novelty detection [21]. The principle of a one-class SVM is to capture regions in the input space where samples form high-density clusters as shown in Figure 3(b). The capability of detecting unseen scan-based attacks will be evaluated for both a one- and two-class SVM. In addition, other learning algorithms, including a neural network and k-nearest-neighbors, are evaluated.

3) Delayed Labeling: Since the overlap in Figure 2 may cause false positives and/or false negatives, behavior is not labeled legitimate or illegitimate in a per-instruction manner; instead, the labeling occurs after every n_dly instructions are executed. Specifically, the behavior is labeled malicious only when at least n_th out of n_dly predictions indicate the existence of an attacker. The optimal values for n_th and n_dly are determined empirically from experiments.

Although mitigated by delayed labeling, false positives and false negatives may still occur. To deal with false positives (i.e., an authorized user is identified as an attacker), the user should have the capability to "reset" the system in order to re-establish access to the JTAG. However, the details of system reset and access re-establishment are beyond the scope of this paper.

III. SVM Hardware Architecture

Ideally, detecting a JTAG attack should be accomplished in real time, at the time a JTAG instruction is loaded. Accomplishing real-time detection therefore requires hardware architecture implementation of an SVM. As shown in Figure 4, the SVM classifier employs a pipelined architecture. The computation of the RBF function uses a look-up table (named RBF-LUT) that is built by calculating all possible values for the operators in advance. In addition, the L1-norm is used for computing the distance between samples. Let n be the length of an opcode sequence, m be the number of bits used for a JTAG opcode, and b be the precision of an RBF result. The memory size of the RBF-LUT is $2^m \times n \times b$. In this work, the memory size is $2KB$ assuming n=4, b=16 and m=8 (n=4 is

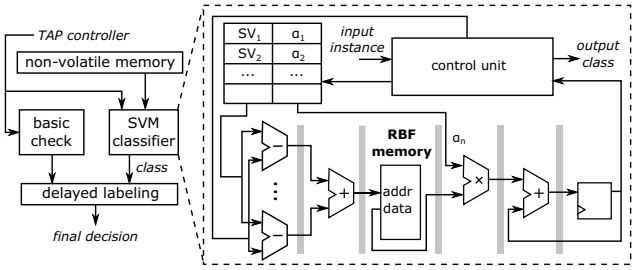


Figure 4: The architecture for detecting scan-based attacks. The SVM classifier employs a pipelined architecture.

an optimal value based on the experiments in Section V, and m=8 stems from the OpenSPARC T2 design). The latency of making a per-instruction prediction is $(S+T-1)$ cycles, where S denotes the number of support vectors and T denotes the number of pipeline stages.

Table 1 compares the area overhead and the latency of the two-layer scheme with [6]. The comparison shows that the two-layer scheme has more area overhead and longer latency but in Section IV we show that it has much better performance.

Approach	Latency	Area (μm^2)	% of chip area
Two-layer scheme	520	755,592	1.79%
[6]	8	452,253	1.07%

Table 1: The latency and the area overhead of the two-layer SVM-based scheme (n=4) is compared with [6].

The area of the SVM classifier depends on n in two folds. First, n affects the width of the pipeline linearly since it represents the dimension of the data. Second, the size of the RAM storing support vectors (named SV-RAM) relies on n which affects both the number of support vectors (SVs) and the size of each SV. A larger n produces more SVs, e.g., the numbers of SVs produced by n=3, 4, 5, 6 are 763, 844, 901 and 962, respectively. However, because these numbers are between 512 and 1024, the column size remains 1024 (each row stores an SV). Thus, the area of the SV-RAM is linearly dependent on n. As shown previously, the area of the RAM storing the RBF-LUT also depends on n linearly. To summarize, the relative area overhead in terms of n is: 0.75 (n=3), 1 (n=4), 1.25 (n=5), 1.5 (n=6).

IV. Experiment

To validate the effectiveness of the two-layer protection scheme, variances of the JTAG programs used in [6] are created for the OpenSPARC T2. Specifically, the variances are generated by substituting different values for the parameters (such as the number of cycles in run-test/idle and test-logic-reset), and by modifying the sequential orders of JTAG instructions used for reverse engineering. The new data set includes 767 normal programs, containing 125,017 opcode sequences (extracted using a fixed-size sliding window), and 1,092 attacking programs, containing 154,980 opcode sequences.

Targeted components			
C_1	Check basic profile	C_5	L2 cache access
C_2	Clock control	C_6	Logic BIST
C_3	Control register	C_7	Memory BIST
C_4	Electronic fuse	C_8	Shadow scan

Table 2: The JTAG programs are divided into eight categories, each one targeting a different IC component.

Attacking strategies	
S_1	Identify the location of the JTAG ports on chip/board
S_2	Identify the length of instruction register (IR)
S_3	Identify the length of each data register (DR)
S_4	Find the opcodes that are not associated with any DR
S_5	Check if each DR can be captured/updated
S_6	Identify the internal scan chains
S_7	Separate opcodes into bundles based on their associated functions
S_8	Determine the nature (control versus data) of each DR
S_9	Investigate adjacent opcodes for characterizing interactions between them

Table 3: The JTAG programs are divided into nine categories, each one exploiting a different scan-based attacking strategy.

An unseen scan-based attack is one that targets a different IC component, or exploits a new scan-based strategy never encountered by the scheme. Thus, in the first set of the experiments, the attacking programs are divided into eight categories based on the components they target (Table 2), while in the second set, the programs are divided into nine categories based on the strategies they exploit (Table 3). More details concerning Table 2 and Table 3 can be found in [6].

For the first set of experiments, the capability to detect attacks targeting a new IC component is evaluated using different learning algorithms as shown in Table 4. The extracted opcode sequences (n=4) are processed by each approach, and two metrics are evaluated, i.e., the accuracy of identifying normal opcode sequences (acc_nor) and the accuracy of identifying unseen attacks (acc_atk). Here, accuracy is defined as the percentage of correct predictions out of all evaluated opcode sequences (either normal operation or unseen attacks). The first set of approaches involve two-class learning algorithms, namely SVM (using an RBF kernel function[1]), neural network (hidden_layer=1, hidden_neuron=10), k-nearest-neighbor (k-NN, k=3) and the decision tree model proposed in [6]. For these approaches, seven out of eight categories of attacks in Table 2 and all cases of normal operations are simulated using 10-fold cross-validation. The resulting accuracy of identifying normal operation is denoted by acc_nor. The eighth category of the component attack is evaluated as an unseen attack and the accuracy acc_atk is measured using the ten classifiers trained in the cross-validation. The second set of approaches involve one-class learning algorithms, including a one-class SVM (ν=0.1) and a one-class k-NN (radius=2). The one-class k-NN identifies an opcode sequence by searching for the neighboring area (radius=2) centered by the opcode sequence. If at least one training sample resides within the

[1]The RBF kernel is used because it performs better than linear- and polynomial-based kernels.

Metric	Algorithm	(a) IC component targeted by attacks								(b) Strategy exploited by attacks						Overall
		C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	S_4	S_5	S_6	S_7	S_8	S_9	
acc_nor	SVM	0.876	0.885	0.876	0.874	0.896	0.876	0.878	0.875	0.876	0.903	0.868	0.886	0.879	0.869	0.878
	Neural net	0.608	0.562	0.473	0.62	0.308	0.604	0.431	0.547	0.753	0.888	0.198	0.528	0.57	0.504	0.51
	k-NN	**0.954**	0.944	**0.97**	**0.956**	0.949	**0.968**	0.945	0.944	**0.971**	**0.956**	0.928	0.935	0.92	0.924	**0.958**
	[6]	0.833	0.923	0.951	0.812	0.912	0.788	0.899	0.912	0.954	0.902	0.924	0.919	0.893	0.834	0.9
	1-class k-NN	0.95	**0.95**	0.95	0.95	**0.95**	0.95	**0.95**	**0.95**	0.946	0.946	**0.946**	**0.946**	**0.946**	**0.946**	0.95
	1-class SVM	0.676	0.676	0.676	0.676	0.676	0.676	0.676	0.676	0.651	0.651	0.651	0.651	0.651	0.651	0.676
	Match	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71
acc_atk	SVM	0.77	0.855	**0.985**	**0.97**	**1**	0.907	0.845	**1**	0.877	**1**	**1**	**0.843**	**0.994**	**0.986**	0.911
	Neural net	**0.934**	**0.876**	0.977	0.954	0.982	**0.91**	0.829	0.999	0.801	0.988	0.998	0.659	0.85	0.918	**0.926**
	k-NN	0.028	0.09	0.379	0.056	0.304	0.161	0.25	0.95	0.521	0.303	0.515	0.689	0.573	0.852	0.235
	[6]	0.91	0.213	0.499	0.439	0.319	0.249	0.13	0.105	**1**	0.94	**1**	0.43	0.47	0.64	0.413
	1-class k-NN	0.574	0.504	0.685	0.437	0.936	0.514	**0.858**	**1**	0.349	0.945	**1**	0.398	0.789	0.854	0.655
	1-class SVM	0.574	0.504	0.685	0.437	0.936	0.514	**0.858**	**1**	0.447	0.929	**1**	0.376	0.833	0.704	0.655
	Match	0.538	0.751	0.964	0.916	0.993	0.79	0.772	**1**	0.599	0.992	**1**	0.762	0.973	0.942	0.835

Table 4: The accuracy of detecting normal operation and unseen attacks that (a) target different IC components and (b) exploit different strategies are evaluated for different learning algorithms (n=4). (Each column with C_i has target component C_i of Table 2 as the unseen attack, i.e., excluded during training. Each column with S_i exploits strategy S_i of Table 3 as the unseen attack, i.e., excluded during training.) The highest accuracy for each set of comparison is in bold.

area, the opcode sequence is then labeled normal. For these approaches, 90% of the normal opcode sequences are used for training, while the remaining 10% are used for evaluating acc_nor. In addition, each category of the component attack is evaluated as an unseen attack. The next approach, named "match", simply uses normal opcode sequences as legal ones and rejects any opcode sequence that does not conform. It is evaluated using similar procedures with one-class learning algorithms, i.e., building a legal database that contains 90% of all cases of normal operation, while using the remaining 10% for measuring acc_nor. Also, each category of the attack is evaluated as an unseen attack.

As shown in Table 4, an SVM has balanced performance for identifying both normal operation and unseen attacks, indicating that the SVM can not only tolerate the variances occurring during normal operation but also detect unseen scan-based attacks. Among the correctly-classified unseen attacks, 0.2% are detected by Layer-I while 99.8% are classified by Layer-II. A neural network and k-NN are capable of identifying normal operation or an attack, but not both. The overall performance of a one-class SVM and a one-class k-NN is worse than an SVM, even for identifying unseen attacks. A likely explanation for this outcome difference is that having some general knowledge of attacks is quite beneficial for detecting new types of attacks. The "match" approach fails to identify variances exhibited by normal operation because the matching rule, by definition, simply does not allow for any variance. Finally, the decision tree model in [6] performs much worse in identifying unseen attacks than the SVM (i.e., 41.3% versus 91.1% overall). However, it is noted that the decision tree performs better than the SVM for identifying attacks targeting C_1. A likely explanation is that "checking the basic profile" of the JTAG may result in frequent easy-to-detect mistakes because the attacker has no prior knowledge of the specific JTAG. The approach in [6] performs better in that situation because various features, other than only the sequential order of instructions, are used to capture user behavior.

For the second set of experiments, the capability to detect attacks exploiting a different strategy is evaluated in a similar manner as shown in Table 4. It is noted that strategies S_1-S_3 of Table 3 are excluded because they can easily be detected by layer-I. The results again show that the SVM has better overall performance than other approaches. It is noted that the decision tree performs better than the SVM for strategy S_4 due to the reasoning earlier described.

Figure 5 shows that the accuracy of identifying normal operation and unseen attacks trends in opposite directions when the opcode sequence length increases. The accuracy of Figure 5 is the average accuracy of each category of attacks weighted by the population size. Figure 5 reveals that a larger n is preferable for detecting unseen attacks, but at the cost of more false positives. The optimal value for n is 4 or 5 if false positives and false negatives have equal cost. Figure 5 also demonstrates that the length of most normal JTAG operations is typically less than 7 or 8.

Figure 6 shows the accuracy after applying delayed labeling for normal operation and unseen attacks. Even though the optimal values for n_dly and n_th should correspond to the highest overall accuracy, n_dly should not be too large because attackers can disguise malicious operations within a long interval more easily. Finally, n_dly=5 and n_th=3 are selected, which achieve an overall accuracy of 94%.

V. Discussion

Table 5 compares the proposed two-layer scheme and other JTAG protection techniques. When compared to encryption techniques, the two-layer scheme is an orthogonal approach that can be combined with encryption to achieve complementing protection for the JTAG. In addition, the proposed scheme is considered valid under the assumptions that the attacker can only access the JTAG ports and does not know how to operate the private JTAG functions.

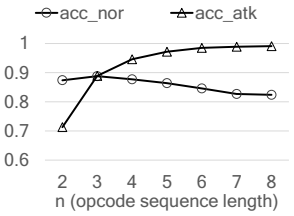


Figure 5: The detection accuracy (y-axis), as a function of opcode sequence length (x-axis), is evaluated for normal operation and unseen attacks.

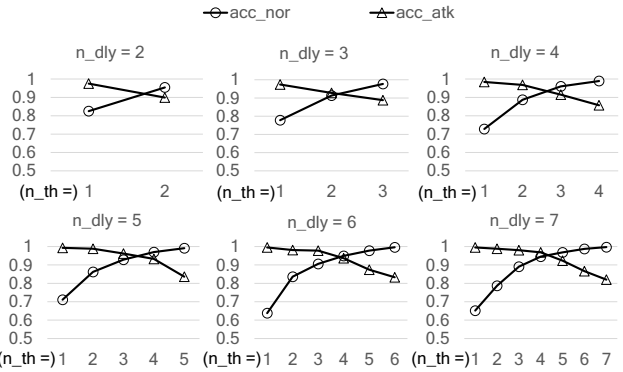


Figure 6: The detection accuracy (y-axis) after applying delayed labeling, in terms of n_dly and n_th, is evaluated for normal operation and unseen attacks.

Protection techniques	Vulnerability	Security level	Over-head
Encryption	Key leakage	High	Medium
Signature detector	False negatives; not valid for unseen attacks	Low	Medium
Anomaly detector	False positives	Medium	Medium
SLIC-J in [6]	Know private JTAG functions; not valid for unseen scan-based attacks	Medium	Medium
Two-layer scheme	Know private JTAG functions	High	Medium

Table 5: Comparison between the proposed two-layer scheme and other JTAG protection techniques.

Further, even if the attacker knows how to operate a portion of the JTAG functions (e.g., the public instructions), the scheme is still effective. Specifically, to reverse other JTAG functions, the attacker will attempt to obfuscate malicious behavior by interleaving intrusions with already-known normal sequences. However, jumps between the intrusions and legitimate operations make the attack easier to identify. This claim is supported by the simulation results. Specifically, assuming that the attacker knows how to operate the control registers (as shown in Table 2), control register operations (with opcode length = 2 to 9) are interleaved with attacks that target unknown components (with opcode length = 1 to 3). In this case, the detection accuracy is 99.9%, which is even higher than the case of no prior knowledge (97.6%).

VI. Conclusion

Detecting attacks based on the user behavior has become a potential technique to ensure the security of modern integrated systems. In this paper, a two-layer learning-based protection scheme is proposed to detect scan-based JTAG attacks. The experiments on the OpenSPARC T2 show that the scheme can detect unseen scan-based attacks with high accuracy (i.e., 94%) under the assumptions that the user can utilize the JTAG ports to apply unseen scan-based attacks.

Acknowledgment

The authors acknowledge the support of the Foundation for Science and Technology of Portugal under Grant BD/28163/2006 (project reference CMU-PT/SIA/0005/2009).

References

[1] F. Domke, "Blackbox JTAG Reverse Engineering," Tech. Rep., 2009.
[2] I. Breeuwsma, "Forensic Imaging of Embedded Systems Using JTAG (Boundary-Scan)," *Digital Investigation*, vol. 3, no. 1, pp. 32–42, 2006.
[3] Z. Basnight, J. Butts, J. Lopez, and T. Dube, "Firmware Modification Attacks on Programmable Logic Controllers," *Critical Infrastructure Protection*, vol. 6, no. 2, pp. 76–84, 2013.
[4] M. Bushnell and V. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-signal VLSI Circuits*. Springer, 2000.
[5] "OpenSPARC T2," http://www.oracle.com/technetwork/systems/opensparc/opensparc-t2-page-1446157.html, Oracle.
[6] X. Ren, V. G. Tavares, and R. D. Blanton, "Detection of Illegitimate Access to JTAG via Statistical Learning in Chip," in *Design, Automation and Test in Europe*, 2015.
[7] J. Da Rolt, A. Das, G. Di Natale, M. Flottes, B. Rouzeyre, and I. Verbauwhede, "Test Versus Security: Past and Present," *Emerging Topics in Computing*, vol. 2, no. 1, pp. 50–62, 2014.
[8] "High Quality Test Solutions for Secure Applications," Mentor Graphics, Tech. Rep., 2010.
[9] B. Yang, K. Wu, and R. Karri, "Scan-based Side-channel Attack on Dedicated Hardware Implementations of Data Encryption Standard," in *International Test Conference*, 2004.
[10] F. Novak and A. Biasizzo, "Security Extension for IEEE Std 1149.1," *Journal of Electronic Testing*, vol. 22, no. 3, pp. 301–303, 2006.
[11] A. Das and U. Kocaba, "PUF-based Secure Test Wrapper Design for Cryptographic SoC Testing," in *Design, Automation and Test in Europe*, 2012.
[12] K. Rosenfeld and R. Karri, "Attacks and Defenses for JTAG," *Design & Test of Computers*, vol. 27, no. 1, pp. 36–47, 2010.
[13] J. Lee, M. Tehranipoor, C. Patel, and J. Plusquellic, "Securing Designs Against Scan-based Side-channel Attacks," *Dependable and Secure Computing*, vol. 4, no. 4, pp. 325–336, 2007.
[14] S. S. Ali, O. Sinanoglu, S. M. Saeed, and R. Karri, "New Scan-based Attack Using Only the Test Mode," in *International Conference on Very Large Scale Integration*, no. 1, 2013.
[15] J. Dworak and A. Crouch, "Don't Forget to Lock Your SIB: Hiding Instruments Using P16871," in *International Test Conference*, 2013.
[16] S. Axelsson, "Intrusion Detection Systems: A Survey and Taxonomy," Technical Report Chalmers University of Technology, Goteborg, Sweden, Tech. Rep., 2000.
[17] I. Yoo, "Protocol Anomaly Detection and Verification," in *International Workshop on Information Assurance*, 2004.
[18] J. M. Estevez-Tapiador, P. Garcia-Teodoro, and J. E. Diaz-Verdejo, "Stochastic Protocol Modeling for Anomaly Based Network Intrusion Detection," in *International Workshop on Information Assurance*, 2003.
[19] M. Ringnér, "What Is Principal Component Analysis?" *Nature biotechnology*, vol. 26, no. 3, pp. 303–304, 2008.
[20] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
[21] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support Vector Method for Novelty Detection," in *Neural Information Processing Systems*, 1999.