ETSI TR 104 066 V1.1.1 (2024-07)

Securing Artificial Intelligence;
Security Testing of AI

Reference

DTR/SAI-0015

Keywords

artificial intelligence, security

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Important notice

The present document can be downloaded from:
https://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

If you find a security vulnerability in the present document, please report it through our Coordinated Vulnerability Disclosure Program:
https://www.etsi.org/standards/coordinated-vulnerability-disclosure

Notice of disclaimer & limitation of liability

The information provided in the present deliverable is directed solely to professionals who have the appropriate degree of experience to understand and interpret its content in accordance with generally accepted engineering or other professional standard and applicable regulations.
No recommendation as to products and services or vendors is made or should be implied.
No representation or warranty is made that this deliverable is technically accurate or sufficient or conforms to any law and/or governmental rule and/or regulation and further, no representation or warranty is made of merchantability or fitness for any particular purpose or against infringement of intellectual property rights.
In no event shall ETSI be held liable for loss of profits or any other incidental or consequential damages.

Any software contained in this deliverable is provided "AS IS" with no warranties, express or implied, including but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights and ETSI shall not be held liable in any event for any damages whatsoever (including, without limitation, damages for loss of profits, business interruption, loss of information, or any other pecuniary loss) arising out of or related to the use of or inability to use the software.

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Securing Artificial Intelligence (SAI).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

Security testing of AI aims at identifying vulnerabilities in AI models. On the one hand, security testing of AI has some commonalities with security testing of traditional software systems. On the other hand, the functioning of AI and in particular ML poses new challenges and requires different approaches for several reasons:

- There are significant differences between symbolic AI, sub symbolic AI, i.e. ML, versus traditional software systems that have strong implications on AI and ML security and on how to test their security properties.
- Non-determinism: AI-based systems can evolve at runtime (self-learning systems), and thus, security properties can degrade at runtime, too. If faced with the same input at different times, self-learning AI-based systems can provide different predictions.
- Test oracle problem: assigning a test verdict is different and more difficult for AI-based systems since not all expected results are known a priori.

- Data-driven algorithms: in contrast to traditional systems, (training) data forms the behaviour of sub symbolic AI, meaning security testing should be extended from the AI component to the data used for training or continuous learning of a system.

Testing consists of several activities that include test planning and control, test design, test implementation, test execution and test evaluation. The present document covers the testing activities test design, test execution and test evaluation. For that purpose, the present document introduces methods and metrics to design test cases (see clause 4), to measure the progress (see clause 5) and to evaluate test cases (see clause 6).

The present document addresses security testing approaches for AI, security test oracles for AI, and definition of test adequacy criteria for security testing of AI. Techniques of each of these topics are applied together to security test a ML component. Security testing approaches are used to generate test cases that are executed against the ML component. Security test oracles enable to calculate a test verdict to determine if a test case has passed, that is, no vulnerability has been detected, or failed, that is a vulnerability has been identified. Test adequacy criteria are used to determine the entire progress and can be employed to specify a stop condition for security testing.

The security testing approaches addressed by the present document are not solely related to security but to robustness as well. Issues with the robustness of ML components can result in both security and safety issues. Security issues of a ML component can enable an adversary to achieve a violation of one of the security properties, i.e. confidentiality, integrity, and availability. Safety issues of a ML component might endanger the environment in which the ML component and the system it is part of is operating. Security issues might also lead to safety issues when, for instance, the availability or integrity of safety measures is affected. Testing of robustness of ML components related to safety-issues in the Automotive domain has been discussed, for instance, in [i.1].

1 Scope

The present document identifies methods and techniques that are appropriate for security testing of ML-based components. Security testing of AI does not end at the component level. As for testing of traditional software, the integration with other components of a system needs to be tested as well. However, integration testing is not the subject of the present document.

The present document addresses:

- security testing approaches for AI;
- security test oracles for AI;
- definition of test adequacy criteria for security testing of AI.

Techniques of each of these topics should be applied together to security test of a ML component. Security testing approaches are used to generate test cases that are executed against the ML component. Security test oracles enable to calculate a test verdict to determine if a test case has passed, that is, no vulnerability has been detected, or failed, that is a vulnerability has been identified. Test adequacy criteria are used to determine the entire progress and can be employed to specify a stop condition for security testing.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] Berghoff, C., Bielik, P., Neu, M., Tsankov, P., & von Twickel, A. (2021, June). Robustness Testing of AI Systems: A Case Study for Traffic Sign Recognition. In IFIP International Conference on Artificial Intelligence Applications and Innovations (pp. 256-267). Springer, Cham.

[i.2] American fuzzy lop.

[i.3] LLVM compiler infrastructure: "libFUZZER".

[i.4] Odena, A., & Goodfellow, I. (2018). Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. arXiv preprint arXiv:1807.10875.

[i.5] Chen, T. Y., Cheung, S. C., & Yiu, S. M. (2020). Metamorphic testing: a new approach for generating next test cases. arXiv preprint arXiv:2002.12543.

[i.6] McKeeman, W. M. (1998). Differential testing for software. Digital Technical Journal, 10(1), 100-107.

[i.7] Pei, K., Cao, Y., Yang, J., & Jana, S. (2017, October). Deepxplore: Automated whitebox testing of deep learning systems. In proceedings of the 26th Symposium on Operating Systems Principles (pp. 1-18).

[i.8] Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., & Madry, A. (2019). Adversarial examples are not bugs, they are features. Advances in neural information processing systems.

[i.9] Tramèr, F., Papernot, N., Goodfellow, I., Boneh, D., & McDaniel, P. (2017). The space of transferable adversarial examples. arXiv preprint arXiv:1704.03453.

[i.10] Demontis, A., Melis, M., Pintor, M., Jagielski, M., Biggio, B., Oprea, A., & Roli, F. (2019). Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In 28th USENIX security symposium (USENIX security 19) (pp. 321-338).

[i.11] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199.

[i.12] Jang, U., Wu, X., & Jha, S. (2017, December). Objective metrics and gradient descent algorithms for adversarial examples in machine learning. In Proceedings of the 33rd Annual Computer Security Applications Conference (pp. 262-277).

[i.13] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

[i.14] Kurakin, A., Goodfellow, I., & Bengio, S. (2016). Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533.

[i.15] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083.

[i.16] Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., & Li, J. (2018). Boosting adversarial attacks with momentum. In Proceedings of the IEEE™ conference on computer vision and pattern recogntion (pp. 9185-9193).

[i.17] Xie, C., Zhang, Z., Zhou, Y., Bai, S., Wang, J., Ren, Z., & Yuille, A. L. (2019). Improving transferability of adversarial examples with input diversity. In Proceedings of the IEEE™/CVF Conference on Computer Vision and Pattern Recognition (pp. 2730-2739).

[i.18] Croce, F., & Hein, M. (2020, November). Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In International conference on machine learning (pp. 2206-2216). PMLR.

[i.19] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A. (2016, March). The limitations of deep learning in adversarial settings. In 2016 IEEE™ European symposium on security and privacy (EuroS&P) (pp. 372-387). IEEE™.

[i.20] Loison, A., Combey, T., & Hajri, H. (2020). Probabilistic Jacobian-based Saliency Maps Attacks. arXiv preprint arXiv:2007.06032.

[i.21] Carlini, N., & Wagner, D. (2017, May). Towards evaluating the robustness of neural networks. In 2017 IEEE™ symposium on security and privacy (sp) (pp. 39-57). IEEE™.

[i.22] Moosavi-Dezfooli, S. M., Fawzi, A., & Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In Proceedings of the IEEE™ conference on computer vision and pattern recognition (pp. 25).

[i.23] Ghiasi, A., Shafahi, A., & Goldstein, T. (2020). Breaking certified defenses: Semantic adversarial examples with spoofed robustness certificates. arXiv preprint arXiv:2003.08937.

[i.24] Croce, F., & Hein, M. (2020, November). Minimally distorted adversarial examples with a fast adaptive boundary attack. In International Conference on Machine Learning (pp. 2196-2205). PMLR.

[i.25] Xiao, C., Zhu, J. Y., Li, B., He, W., Liu, M., & Song, D. (2018). Spatially transformed adversarial examples. arXiv preprint arXiv:1801.02612.

[i.26] Mopuri, K. R., Garg, U., & Babu, R. V. (2017). Fast feature fool: A data independent approach to universal adversarial perturbations. arXiv preprint arXiv:1707.05572.

[i.27] Hayes, J., & Danezis, G. (2018, May). Learning universal adversarial perturbations with generative models. In 2018 IEEE™ Security and Privacy Workshops (SPW) (pp. 43-49). IEEE™.

[i.28] Chen, P. Y., Zhang, H., Sharma, Y., Yi, J., & Hsieh, C. J. (2017, November). Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (pp. 15-26).

[i.29] Brendel, W., Rauber, J., & Bethge, M. (2017). Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. arXiv preprint arXiv:1712.04248.

[i.30] Andriushchenko, M., Croce, F., Flammarion, N. & Hein, M. (2020, August). Square attack: a query-efficient black-box adversarial attack via random search. In European Conference on Computer Vision (pp. 484-501). Springer, Cham.

[i.31] Uesato, J., O'donoghue, B., Kohli, P., & Oord, A. (2018, July). Adversarial risk and the dangers of evaluating against weak attacks. In International Conference on Machine Learning (pp. 5025-5034). PMLR.

[i.32] Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D. & Kurakin, A. (2019). On evaluating adversarial robustness. arXiv preprint arXiv:1902.06705.

[i.33] Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., & Madry, A. (2018). A rotation and a translation suffice: Fooling CNNs with simple transformations.

[i.34] Alzantot, M., Sharma, Y., Chakraborty, S., Zhang, H., Hsieh, C. J., & Srivastava, M. B. (2019, July). Genattack: Practical black-box attacks with gradient-free optimization. In Proceedings of the Genetic and Evolutionary Computation Conference (pp. 1111-1119).

[i.35] Moosavi-Dezfooli, S. M., Fawzi, A., Fawzi, O., & Frossard, P. (2017). Universal adversarial perturbations. In Proceedings of the IEEE™ conference on computer vision and pattern recognition (pp. 1765-1773).

[i.36] Dong, Y., Zhang, P., Wang, J., Liu, S., Sun, J., Hao, J. & Ting, D. (2019). There is Limited Correlation between Coverage and Robustness for Deep Neural Networks. arXiv preprint arXiv:1911.05904.

[i.37] Zhu, H., Hall, P. A., & May, J. H. (1997). Software unit test coverage and adequacy. Acm computing surveys (csur), 29(4), 366-427.

[i.38] Kim, J., Feldt, R., & Yoo, S. (2019, May). Guiding deep learning system testing using surprise adequacy. In 2019 IEEE™/ACM 41st International Conference on Software Engineering (ICSE) (pp. 1039-1049). IEEE™.

[i.39] Ma, L., Juefei-Xu, F., Zhang, F., Sun, J., Xue, M., Li, B. & Zhao, J. (2018, September). Deepgauge: Multi-granularity testing criteria for deep learning systems. In Proceedings of the 33rd ACM/IEEE™ International Conference on Automated Software Engineering (pp. 120-131).

[i.40] Weng, T. W., Zhang, H., Chen, P. Y., Yi, J., Su, D., Gao, Y. & Daniel, L. (2018). Evaluating the robustness of neural networks: An extreme value theory approach. arXiv preprint arXiv:1801.10578.

[i.41] Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., & Usunier, N. (2017). Parseval networks: Improving robustness to adversarial examples. arXiv preprint arXiv:1704.08847.

[i.42] Katz, G., Barrett, C., Dill, D. L., Julian, K., & Kochenderfer, M. J. (2017, July). Reluplex: An efficient SMT solver for verifying deep neural networks. In International Conference on Computer Aided Verification (pp. 97-117). Springer, Cham.

[i.43] Mayer, J., & Guderlei, R. (2004). Test oracles using statistical methods. Testing of component-based systems and software quality.

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

adversarial example: carefully crafted input which mislead a model to give an incorrect prediction

perturbation: semantically meaningless modification of an input

EXAMPLE: Perturbation can have the form of noise added to an image.

substitute model: model created by an adversary to craft transferable adversarial examples

NOTE 1: The substitute model performs the same task as the target model but may use a different ML technique or a different dataset.

NOTE 2: The terms surrogate model and substitute model are used synonymously.

surrogate model: See substitute model.

target label: label that an adversary wants the target model to output if fed with an adversarial example

target model: model an adversary wants to make wrong predictions

transferable adversarial example: adversarial example which is crafted for one model but can also fool a different model with a high probability

true label: correct label for an input from the ground truth

3.2 Symbols

For the purposes of the present document, the following symbols apply:

L_0	Pseudo distance (number of non-zero elements)
L_2	Euclidean distance
L_∞	Chebyshev distance
L_{flow}	Flow field function
L_p	Distance that needs to be specified by the parameter p with $p \in \{0,2,\infty\}$

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AI	Artificial Intelligence
CLEVER	Cross Lipschitz Extreme Value for nEtwork Robustness
DSA	Distance-based Surprise Adequacy
FAB	Fast Adaptive Boundary attack
FGSM	Fast Gradient Sign Method
JSMA	Jacobian-based Saliency Map Attack
L-BFGS	computer-memory-Limited approximation of the Broyden-Fletcher-Goldfarb-Shanno algorithm
LSA	Likelihood-based Surprise Adequacy
ML	Machine Learning
NaN	Not a Number
PGD	Projected Gradient Descent
ReLU	Rectified Linear Unit
SAI	Securing Artificial Intelligence
SPSA	Simultaneous Perturbation Stochastic Approximation
TJSMA	Taylor JSMA
WJSMA	Weighted JSMA

ZOO Zeroth Order Optimization-based attacks

4 Security testing techniques

4.1 Introduction

Security testing techniques are used for designing test cases that are later on executed against an ML component. Such test cases consist of the input data that is fed to the ML component to identify a vulnerability, e.g. a susceptibility to a specific adversarial example. Clause 4 presents different approaches that can be employed for crafting such inputs. The presented testing approaches can be divided into those that have been developed for traditional software and can be employed for security testing of ML components as well, and those that are specific to ML. Furthermore, not all of them are security-specific but can be more versatile with respect to the quality characteristics in question.

NOTE: It is necessary to ensure that the system is not designed to recognise the adversarial examples used in a test environment and to run in such a way that the test is passed by bypassing normal operation.

4.2 Mutation testing

4.2.1 Coverage-guided fuzzing

Coverage-guided fuzzing is a technique that has been established for traditional software systems. For such systems, code coverage has been extensively used as coverage metrics together with genetic algorithms, mostly using binary mutation without protocol models, as in American Fuzzy Lop [i.2] and libFuzzer [i.3]. Odena et al. [i.4] transferred this approach to neural networks of different architectures. Instead of random binary mutation, they use specific mutators for images and text. For images, their approach mutates existing pictures by adding white noise either to the extent of a user-configurable variance or by a user-configurable L_∞ norm. As distance metric the approximate nearest neighbour that is greater than a given threshold is used and assume a higher coverage is the distance to the nearest neighbour is above this threshold.

NOTE: L_∞ norm or Chebyshev distance simply takes the (mathematically absolutely) largest component of a vector.

4.2.2 Metamorphic testing

Metamorphic testing [i.5] is a testing approach that relies on metamorphic relations to identify test inputs for which the relationships between their outputs are known or could be identified, for instance using statistical methods. Based on existing, passing test cases, new test cases can be derived using the metamorphic relations. Hence, metamorphic testing requires the identification of metamorphic relations as a first step. This can be a challenging task for complex scenarios where relationships between different inputs and output are not obvious. The simplest example of a metamorphic relation is for the sine function where two metamorphic relations can be derived from the periodicity of the sine function:

$$\sin x = \sin(x + 2\pi) \quad (1)$$

and

$$|\sin x| = |\sin(x + \pi)| \quad (2)$$

Metamorphic relations can be more complex than simple equality and the absolute value and can involve any mathematical function. They are usually specific to the problem domain.

4.3 Differential testing

Differential testing [i.6] is a testing technique developed for traditional software that uses another system as a reference system to identify deviations of the system under test when different behaviours of both systems can be observed. Test cases are generated randomly, and test cases that result in different behaviours between the system under test and the reference system are considered to have revealed a bug and are retained as regression test and for debugging purposes.