Project 3 Handout

Machine Learning in Cybersecurity (WS 2020)
v1.0

1. Overview

In this project, you will be working on your own idea by trying out something new or doing a study. We will help you along the process by providing intermediate feedback. This hopefully leads you to contribute towards the rapidly growing field of machine learning in cybersecurity!

The following document provides logistic information and pointers to get you started.

2. Logistics

In this section, we provide administrative details specific to the project, such as timelines, format of report and presentation, etc.

2.1. Timeline

- **December 14, 2020**: Project handed-out
- **December 21, 2020 (13:59)**: Submit proposals to tutors
- **December 25, 2020**: Receive feedback from tutors
- **January 11, 2021***: Mid-term presentation
- **February 2, 2021 (23:59)***: Project report due
- **TBD, 2021***: Project presentations (open to everyone, exact timeslots TBD)

*: tentative

2.2. Submitting Proposals

The proposal you submit to your tutors should be structured as:

1. **Problem Statement**: What will you work on?
2. **Motivation**: Why this particular problem?
3. **Proposed Strategy**: How do you plan to approach it?
4. **Related work**: What's been done so far?
5. **Existing code/software**: What's available to get you started?
6. **Implementation**: What needs to be implemented?
7. **Evaluation - metrics**: Which metrics will you use to evaluate your idea?
8. **Evaluation - datasets**: On which datasets (real/synthetic) will you perform evaluation?
9. **Evaluation - baselines**: Against which methods will your compare your results?
10. **Success criteria**: What results do you need to consider the project successful?
11. **Team**: Team name and names of members

Please address each of these *succinctly* (at most 2 lines) and email it as a PDF (at most 1 page) to the tutor mailing list (mlcysec_ws2021_staff@lists.cispa.saarland) by December 21. We will provide feedback before Christmas break.

We strongly suggest being *specific* and additionally, investing plenty of time in your proposal. Based on experience, we find the quality of feedback proportional to the effort invested in it. While writing this proposal, also take into account practical factors such as whether it can be easily contributed by multiple members, the time to setup/run experiments, vacations, etc.

2.3. Project Report

We require you to hand in a report (details below) where you describe your project and the results you obtain. Analogous to a short research paper, it should be structured with an abstract, an introduction, background, a description of your experimental setting and results and finish with a conclusion. Do not forget to add all sources you found!

Using LaTeX and the IEEE double column format[1] (currently used in this document), you should submit at most **four pages** excluding references – a standard format for workshop papers in conferences. Please do *not* alter attributes (font size, margins, line spacing) in the template – this allows us to fairly judge all submissions. We recommend using Overleaf[2] to write the paper as it allows for collaboration among team members.

Keep in mind that you are graded for content, not for length: an on the point description of a project on two pages earns you a better grade than a four page repetitive one!

Discussion of results plays a crucial role. Do not simply report numbers and graphs, and hope the reader reaches the same conclusions you did (they usually don't). Always, follow it up with a discussion – what result did you expect? what did you find? why is this suprising? The converse

[1] http://cvpr2021.thecvf.com/sites/default/files/2020-09/cvpr2021AuthorKit_2.zip

[2] https://www.overleaf.com

holds too. If you are discussing an observation, it must be substantiated with results or references. For instance, if you claim "X and Y improves performance of method M", it must be backed-up – either in figures, tables, or inline – with performance of $(\neg X, \neg Y), (X, \neg Y), (\neg X, Y)$.

2.4. Project Presentations

In contrast to your previous projects, you will now present your projects additionally in front of your fellow students. As soon as we have the project topics, we will inform you about the presentation schedule.

Note that the project presentation *complements* your project report. We do not advise reiterating every line, plot, and table from your report in the presentation. Due to the limited amount of time, this strategy obscures your story and findings. Rather, use the minimal information set necessary to clearly convey a strong story. We will read your report to fill-in the gaps.

The tentative plan is to have $\sim$10 minutes presentation followed by Q&A. We will strictly enforce this time limit, by penalizing presentations that run over time. Please mail us the slides at least an hour before your presentation.

2.5. Implementation and Experiments

Unlike previous projects, you are free to choose language, libraries and framework. For the deep learning libraries, we recommend sticking to the popular ones (TensorFlow, PyTorch) since they are well-documented and issues you run into will highly likely be discussed in the forums. While you do not have to submit the code and models, please keep it accessible in case we need to take a look.

2.6. Honor Code

We encourage discussing ideas and concepts with other students to help you learn and better understand the course content. However, the work you submit and present **must be original** and demonstrate your effort in solving the presented problems. We will not tolerate blatantly using existing solutions (such as from the internet), improper collaboration (e.g., sharing code or experimental data between groups) and plagiarism. Specifically for the report, clearly cite your sources. If the honor code is not met, no points will be awarded.

3. Getting Started

In this section, we provide pointers to get you started. Although the lists that follow are not exhaustive, our intention is to simply point you towards the "rabbit holes".

Our suggestion is to first narrow down your interests based on the course lectures. Alternatively, understanding the landscape by reading survey papers [9] or looking at AI+ML tracks in conferences or workshops. Following this, pick few specific papers that intrigue you and figure out if you can build on top of it. Repeat these steps until you're satisfied.

3.1. Reading

Here we provide a list of interesting survey papers and top-tier venues where relevant research in this field is published.

- Survey Papers on Adversarial Machine Learning [2, 9]
- Comprehensive Analysis of Inference Attacks [7]
- Relevant conferences
  - Privacy and Security: S&P (Oakland), CCS, USENIX, NDSS
  - Machine Learning: NIPS, ICLR, ICML, AAAI
  - Computer Vision: CVPR, ICCV, ECCV
  - NLP: EMNLP
- Relevant AI+ML workshops: PPML (at NIPS), SecML (at NIPS), COPS (at CVPR), AISec (at CCS)
- Curated link collection
  - `https://github.com/jivoi/awesome-ml-for-cybersecurity`
  - `https://github.com/RandomAdversary/Awesome-AI-Security`
  - `https://github.com/shramos/Awesome-Cybersecurity-Datasets`

3.2. Existing Tools

- Adversarial examples: CleverHans[3] or Foolbox[4]
- Pretrained models: Many deep learning libraries already offer them e.g., ResNet in Pytorch[5]
- Differential Privacy: TensorFlow[6]
- Datasets:
  - Classification: Adult Income, Enron-Spam
  - Images: MNIST, CIFAR, GTSRB
  - Note: It is ok generating synthetic datasets, or using subsets of existing datasets. Clearly document how the examples were generated/sampled.

3.3. Ideas

Now we present some example ideas, to help you get a better feeling of potential exciting problems. By no means

[3] `https://github.com/tensorflow/cleverhans`
[4] `https://foolbox.readthedocs.io/en/latest/`
[5] `https://pytorch.org/docs/stable/torchvision/models.html`
[6] `https://github.com/tensorflow/models/tree/master/research/differential_privacy`

is this exhaustive. We strongly encourage you to also look in other directions. We will gladly provide you feedback and additional resources if you let us know what interests you.

Adversarial Examples. Possible projects consist in breaking proposed defenses (as in [1, 3], feel free to contact us for unbroken defenses). You might as well introduce or improve existing attacks, or target particular scenarios or constraints.

Inference Attacks on ML models. Due to the growing popularity and effectiveness of ML, they are increasingly used in numerous applications e.g., photo assistants on your smart-phone, navigation and control autonomous vehicles. Hence, it is crucial to understand the privacy and security of ML models when deployed for such applications. To aid understanding, we ask: "how can an adversary A abuse such deployed ML models?" A common recipe to approach such problems require stating:

- **Assumptions**: What information does A have on the ML model – Is it a black- or white-box? Is the training data known? Is the model family known? ...
- **Adversary's task**: What is A's end goal - stealing the model (see [8, 13])? reconstructing training data (see [4, 6])? inferring members (see [7, 11, 12])?...
- **Attack vector/surface**: How does A interact with the ML model – via input queries? by providing it training data? ...

To contribute along these lines, some suggestions: identify a new task, identify a new attack vector, or perform some task using minimal assumptions. We suggest reading papers [7, 9] to get started in this direction.

Defenses for Inference Attacks. You could introduce novel defense measures to guard against any of the above inference attacks. In this case, first, fix an attack model based on one or more papers (ideally, whose implementations are available). Now, play the role of the victim and identify measures to thwart the adversary's attack. Typically, this is presented as a privacy-utility dilemma wherein victim achieves privacy at some cost to utility (e.g., more computation, lower performance). Although it is difficult to find complete papers which present defenses, it is common for most "attack" papers to also propose countermeasures in the final section. You could potentially build on top of these countermeasures.

Malicious Code Detection Malicious programs especially the script codes in webpages on the Internet is a critical security issue. There are several attempts which try to employ deep-learning-based approaches to detect these malicious programs [5, 10, 14]. Possible project in this direction is breaking these classifiers by devising a method to create adversarial examples for script codes (e.g. obfuscating the malicious scripts).

Changelog

- **v1.0**: First version

References

[1] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[2] B. Biggio and F. Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[3] N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017.

[4] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015.

[5] D. Hendler, S. Kels, and A. Rubin. Detecting malicious powershell commands using deep neural networks. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 187–197. ACM, 2018.

[6] B. Hitaj, G. Ateniese, and F. Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[7] M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks. *arXiv preprint arXiv:1812.00910*, 2018.

[8] T. Orekondy, B. Schiele, and M. Fritz. Knockoff nets: Stealing functionality of black-box models. 2019.

[9] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman. Sok: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 399–414. IEEE, 2018.

[10] G. Rusak, A. Al-Dujaili, and U.-M. O'Reilly. Ast-based deep learning for detecting malicious powershell. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 2276–2278. ACM, 2018.

[11] A. Salem, Y. Zhang, M. Humbert, M. Fritz, and M. Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *NDSS*, 2018.

[12] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *Security and Privacy (SP), 2017 IEEE Symposium on*, 2017.

[13] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart. Stealing machine learning models via prediction apis. In *USENIX Security Symposium*, 2016.

[14] Y. Wang, W.-d. Cai, and P.-c. Wei. A deep learning approach for detecting malicious javascript code. *Security and Communication Networks*, 9(11):1520–1534, 2016.