Mock Exam

Security and Privacy of Machine Learning (NWI-IMC069)

1. **What aspects of the CIA triad are compromised from the attacks on AI (evasion attack, poisoning attack, sponge poisoning attack, backdoor attack, model stealing attack, membership inference attack, and model inversion attack). As a reminder the CIA triad stands for:**

   - ***Confidentiality*: Only authorized parties can access information or services.**
   - ***Integrity*: Information is protected from unauthorized alteration (i.e., creation, modification, and deletion).**
   - ***Availability*: Information or services must be available to all authorized parties whenever they are needed.**

   A short explanation of each choice is required.

   Answer: The evasion attack alters the inputs of a model in a malicious way targeting misclassifications. Thus, it violates the integrity of the model. Poisoning attacks modify the training data to change the deployed model's behavior violating the dataset's integrity. Also, in the untargeted poisoning attack the model's performance is substantially deteriorated violating the model's availability too. In sponge poisoning attack apart from the dataset's integrity the increased memory consumption of the poisoned model leads also to a denial of service, violating the system's availability. In a backdoor attack a part of the training data or the model's weights need to be altered violating the dataset's integrity. Also if the algorithm is altered from an unauthorized party to insert a backdoor confidentiality is also violated. Model stealing attack violates the confidentiality of the system as the adversary steals a deployed model from another party. In membership inference an adversary can infer if a specific data point was used for the model's training violating the model's confidentiality. In model inversion the adversary accesses the data used for the training, violating the dataset's confidentiality.

2. **What threat model shall we assume for the adversary when designing an attack, black box or white box? Why? Similarly, what threat model shall we use for an attacker when designing a defense? Why?**

   Answer: When we design an attack it is better to assume black-box threat model because it uses no assumptions for the adversary. As a result, the attack is more general and if it is successful it has wide applicability. When designing a defense, to ensure that it is effective in many cases we should assume a white box threat model for the potential attacker (adaptive attacker). In this way we are not violating Kerckhoff's principle (security by obscurity).

3. Threat modeling:

   ◯ **Uses models to identify security flaws**.

   ◯ Uses implementation details.

◯ Provides theoretical guarantees for its outcomes.

◯ All of the above.

◯ None of the above.

4. **Explain the concept of federated learning and discuss its potential security vulnerabilities. What are the key challenges in ensuring the security and privacy of data in federated learning systems?**

   Answer: Federated learning is a machine learning approach in which the training process is decentralized, allowing multiple devices or servers to collaboratively build a shared model without sharing raw data. Each device or client independently trains the model using its local data; only the model updates are exchanged.

   However, federated learning introduces several potential security vulnerabilities. One major concern is data privacy. As data remains local, sensitive information can be exposed during the model aggregation or communication process. Adversaries may attempt to extract sensitive information from the model updates or reconstruct the original data, i.e., inference or model inversion attacks.

   Another vulnerability is poisoning attacks. Malicious participants can intentionally inject manipulated data into the training process, compromising the integrity and performance of the global model. These attacks lead to a compromised aggregated model, which will be shared among all clients.

   Ensuring the security and privacy of data in federated learning systems is challenging. One challenge is implementing robust security mechanisms to protect the model updates and prevent unauthorized access or tampering. Encryption techniques, secure aggregation protocols, and access control mechanisms can help mitigate these risks. In order to prevent poisoning attacks, the server could incorporate secure aggregation techniques such as FLAME to discard malicious model updates.

5. **Describe a full threat model for an attack of your choice.**

   Answer: As we discussed today, explain attacker knowledge, capabilities, and goals.

6. **What are the main differences in the training of SL and FL (briefly explain)? What is the main advantage of using SL over HFL? what is the advantage of using Boomerang SL over Vanilla SL?**

   Answer: The first and second answers can be found on slides 20-22 in lecture 12. The third answer is: by keeping the labels on the client's side, the labels are also protected from the server.

7. Can we make an evasion attack in federated learning (FL)?

   ◯ No. Evasion attacks can only be made in centralized ML.

   ◯ Yes. Also, it is different from centralized ML.

   ◯ **Yes. However, the setup is the same as in ML.**

   ◯ No. In FL, only poisoning attacks are possible.

8. **Why do adversarial examples exist?**

   Answer: The standard trained model learns both robust and no-robust features. Robust and no-robust features are helpful for classification when inputs are clean. But no-robust features are highly predictive, yet brittle and (thus) incomprehensible to humans. No-robust features may be used to build imperceptible but strong perturbation that mislead the model.

9. **Describe the FGSM algorithm.**

   Answer: in Lecture 3 slides, from p27.

10. **List 3 defenses against evasion attack and describe one of those.**

    Answer: 1. adversarial training; 2. adversarial pruning; 3. Obfuscated Gradients; 4. Random input transformation.

11. **What is the difference between MIA and AIA privacy attacks?**

    Answer: The goal of MIA is to find the data used to train a machine learning model. The goal of AIA is to infer the unknown values of the sensitive features.

12. **Describe how you need to rewrite the FGSM code below to change it into a PGD attack.**

    - Explain what parameters you would include/remove
    - List which lines of code you would modify
    - Explain what modifications you would perform
    - We do not expect exact code, but make a list with pseudocode and explanations of the code

```python
class FGSM(Attack):

    def __init__(self, model, eps=8/255):
        super().__init__("FGSM", model)
        self.eps = eps
        self.supported_mode = ['default', 'targeted']

    def forward(self, images, labels):
        images = images.clone().detach().to(self.device)
        labels = labels.clone().detach().to(self.device)

        if self.targeted:
            target_labels = self.get_target_label(images, labels)

        loss = nn.CrossEntropyLoss()

        images.requires_grad = True
        outputs = self.get_logits(images)

        # Calculate loss
        if self.targeted:
            cost = -loss(outputs, target_labels)
        else:
            cost = loss(outputs, labels)

        # Update adversarial images
        grad = torch.autograd.grad(cost, images,
                                   retain_graph=False, create_graph=False)[0]

        adv_images = images + self.eps*grad.sign()
        adv_images = torch.clamp(adv_images, min=0, max=1).detach()

        return adv_images
```

Answer:

Parameters to add:

- alpha: (float) PGD step size; used to limit the size of the step in the direction of the sign of the gradient
- steps: (integer) number of steps to take; PGD is a multiple-step approach or iterative version of FGSM
- random_start: (boolean) specifies if PGD should start from a random point.

Parameters to remove: None

Lines to modify:

- After line 15 and before line 17: clone original images into a new variable (lets call it adv_images) which is used during the iterative process.
- After line 15 and before line 17: add check for random_start; if random_start is true then add random uniform values between -eps and eps to the adv_images and afterwards make sure the values are between min=0 and max=1.
- Just before code on line 17: add for loop to iterate the number of steps and make sure lines 17 <–> 31 are inside this for loop
- line 17: change images to adv_images
- line 18: change images to adv_images
- line 27: change images to adv_images
- line 30: change images to adv_images; change self.eps to self.alpha
- directly after line 30: add a new variable delta and define it as adv_images minus images where the values are clamped between min=-self.eps and max=self.eps
- line 31: change second adv_images to images + delta