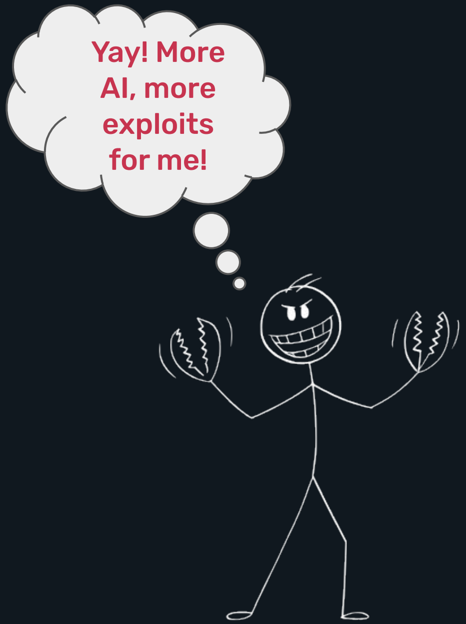
Incubated Machine Learning Exploits: Backdooring ML Pipelines Using Input-Handling Bugs

Suha Sabi Hussain

Who am I?

- Security engineer at ToB
- AI/ML security
- Georgia Tech alumni
- Queens, New York



Source: Kerr, Dara. “Armed with traffic cones, protesters are immobilizing driverless cars.” NPR, 2023

So how can we build our own exploits?

Motivation

A Tale of an Incubated ML Exploit and a Robotics Competition

#1:
Obtain the model file (somehow) 😈

#2:
Inject a model backdoor using a file format RCE

#3:
Watch as the vehicle ignores the stop signs with stickers

Motivation

Source: "BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain" (Gu et al., 2017)

A Tale of an Incubated ML Exploit and a Robotics Competition

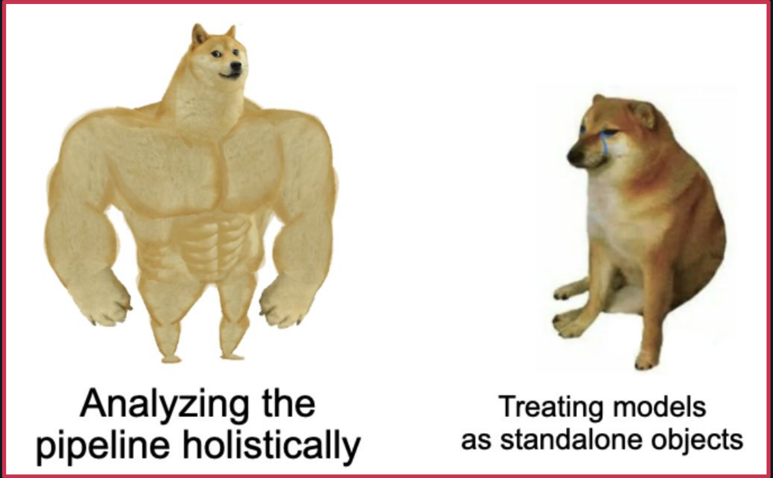
#1:
Obtain the model file (somehow) 😈

#2:
Inject a model backdoor using a file format RCE

#3:
Watch as the vehicle ignores the stop signs with stickers

Exploit frameworks, file formats, backdoors, and more!

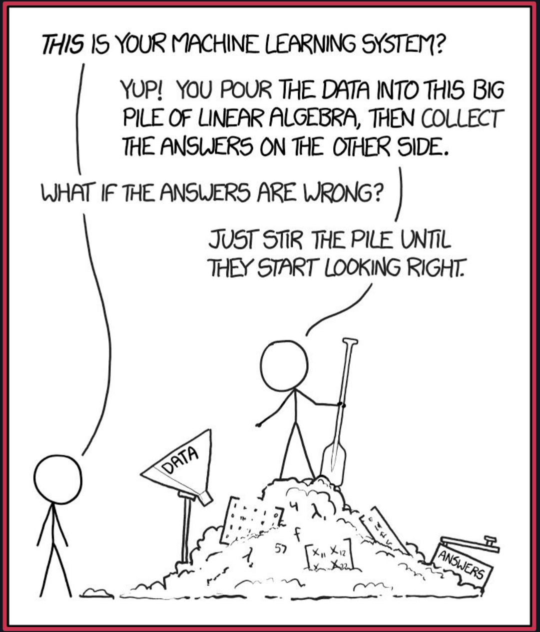
Talk overview

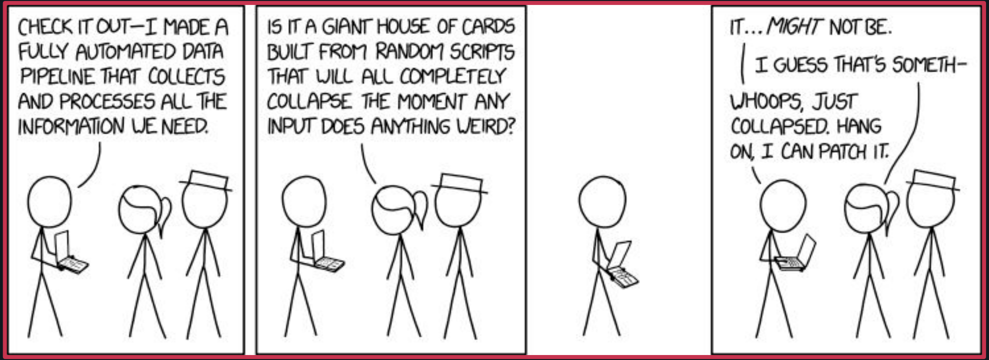


Model vulnerabilities and ML backdoors

All models are wrong; some are useful

Trust no model…



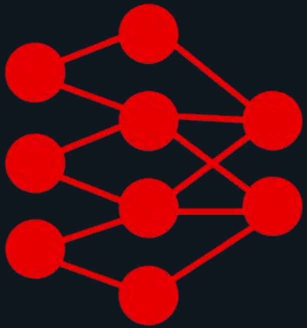


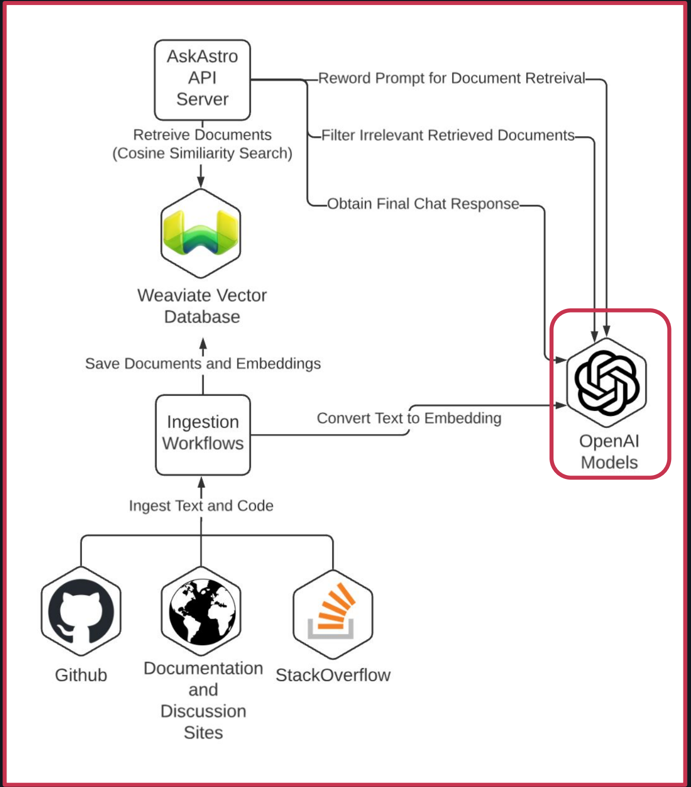
Source: XKCD #1838, #2054

A ML backdoor attack allows a malicious actor to force a model to produce specific outputs given inputs in the presence of an attacker-chosen trigger.

But model vulnerabilities can be difficult to exploit in the real world.

The rest of the system gets ignored!

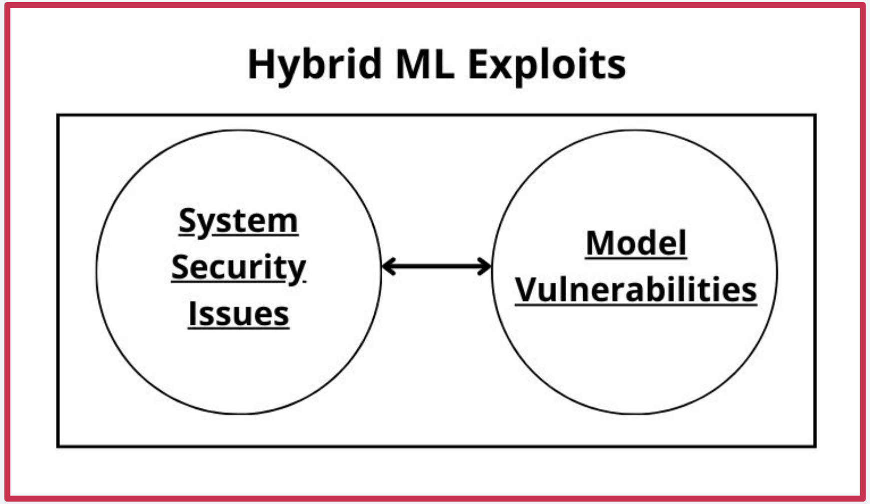
Input →  → Output



Source: “Auditing the Ask Astro LLM Q&A app” (Trail of Bits Blog)

Hybrid and incubated ML exploits

A hybrid ML exploit chains a system security issue with a model vulnerability.



Summoning Demons

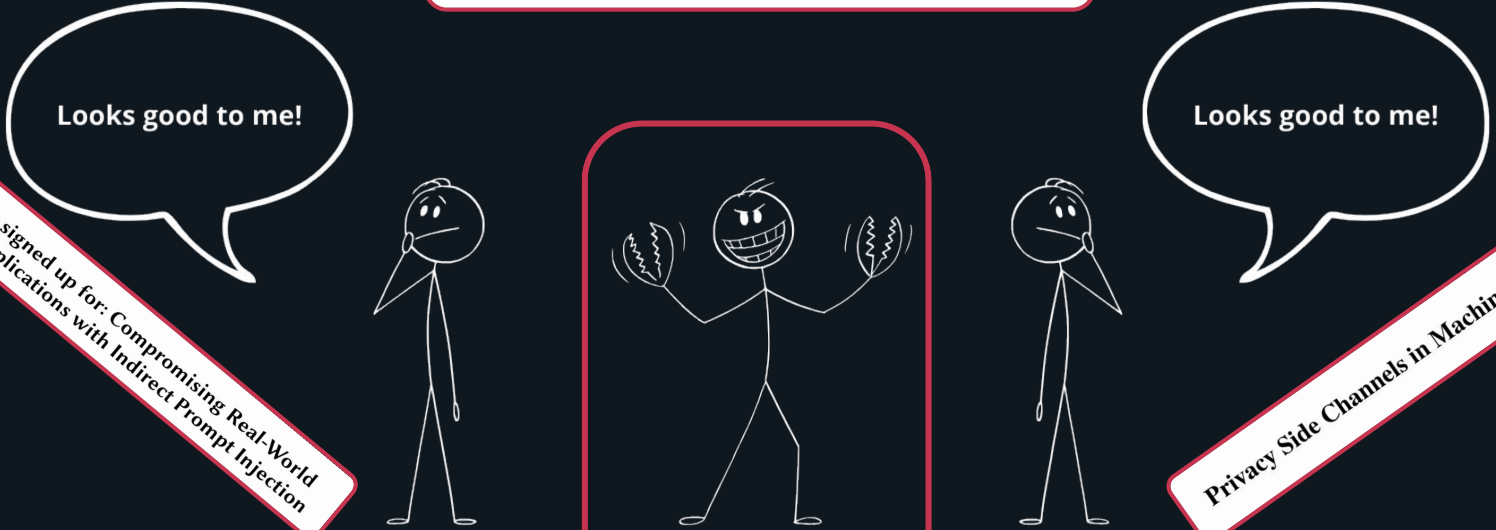
The Pursuit of Exploitable Bugs in Machine Learning

Poisoning Web-Scale Training Datasets is Practical

Learned Systems Security

Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection

Privacy Side Channels in Machine Learning Systems



Model Security

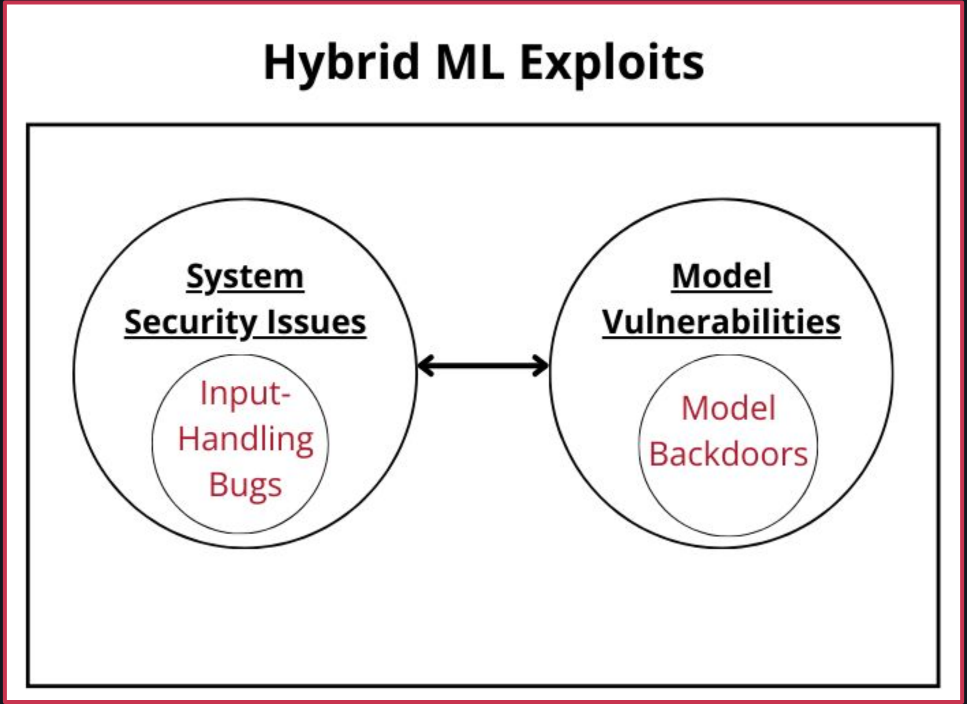
Hybrid ML Exploits

System Security

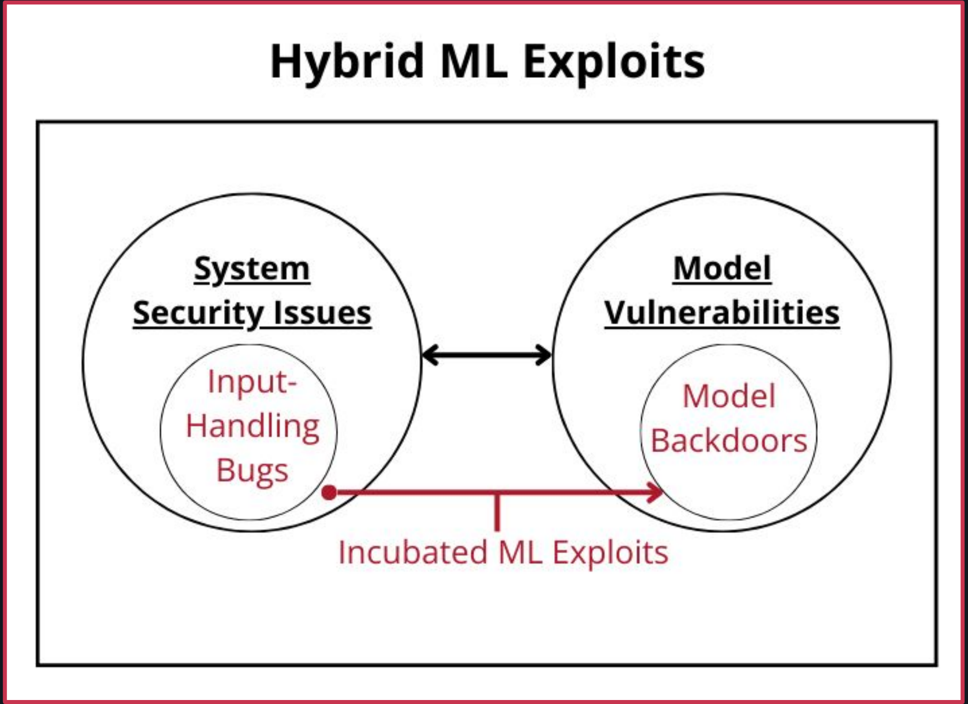
On the Exploitability of Audio Machine Learning Pipelines to Surreptitious Adversarial Examples

Hidden Backdoors in Human-Centric Language Models

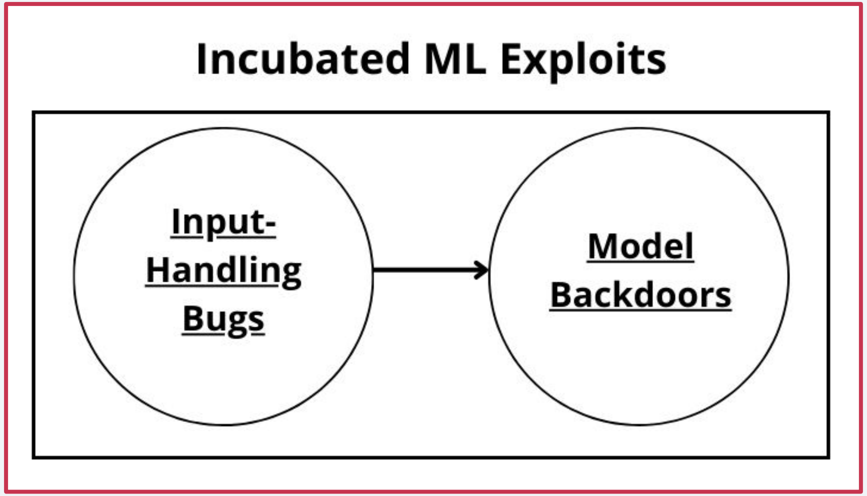
Exploit Cartography



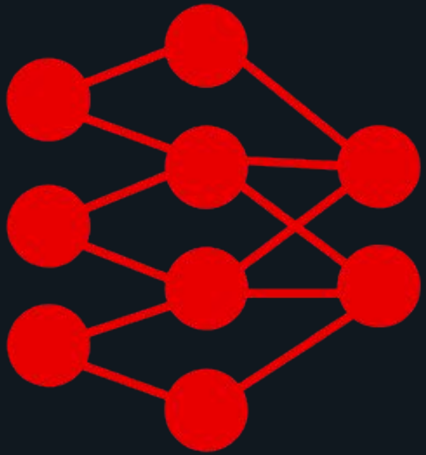
It's dangerous to go alone! Take this.



An incubated ML exploit uses an input-handling bug in the system to inject a backdoor.



Backdoor Injection



Attackers can:

- Change the parameters
- Change the architecture

Input-handling bugs and LangSec

We serialize and deserialize these files

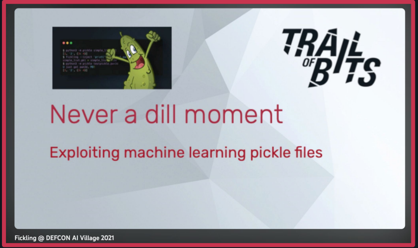
ML models are stored as files!

"…a file has no intrinsic meaning. The meaning of a file—its type, its validity, its contents—can be different for each parser or interpreter."

-Ange Albertini (PoC || GTFO 7:6)

GGUF, the long way around

Data Scientists Targeted by Malicious Hugging Face ML Models with Silent Backdoor



0x36 / weightBufs

ANE kernel r/w exploit for iOS 15 and macOS 12

trailofbits / ml-file-formats

WEAPONIZING ML MODELS WITH RANSOMWARE

EleutherAI, Hugging Face Safetensors Library

Security Assessment

May 3, 2023

Prepared for:
Stella Biderman
EleutherAI
Nicolas Patry
Hugging Face
Garry Jean-Baptiste
Stability AI

Prepared by: **Fredrik Dahlgren, Suha Hussain, Heidy Khlaaf, and Evan Sultanik**

llama.ttf

`llama.ttf` is a font file which is also a large language model and an inference engine for that model.

LangSec

1. Applies formal language theory to security problems
2. Exploitable parser bugs are common
   a. Simple and well-defined inputs
   b. Full validation by a minimalist recognizer
3. All input-handling bugs are the product of “insufficient recognition” or “parser differentials”
   a. Exploit systems using polyglot files or ambiguous files
   b. More info: https://langsec.org/

Non-minimalist input-handling code

Shotgun parsing

Input language too complex

The Seven Turrets of Babel: A Taxonomy of LangSec Errors and How to Expunge Them

Falcon Darkstar Momot
Leviathan Security Group
Seattle, WA
falcon@falconk.rocks

Sergey Bratus
Dartmouth College
Hanover, NH
sergey@cs.dartmouth.edu

Sven M. Hallberg
Hamburg University of Technology
Hamburg, Germany
sven.hallberg@tuhh.de

Meredith L. Patterson
Upstanding Hackers, Inc.
Brussels, Belgium
mlp@upstandinghackers.com

Incomplete protocol specification

Permissive processing of invalid input

Overloaded field in input format

Differing interpretations of input language

The Seven Turrets of Babel: A Taxonomy of LangSec Errors and How to Expunge Them

Falcon Darkstar Momot
Leviathan Security Group
Seattle, WA
falcon@falconk.rocks

Sergey Bratus
Dartmouth College
Hanover, NH
sergey@cs.dartmouth.edu

Sven M. Hallberg
Hamburg University of Technology
Hamburg, Germany
sven.hallberg@tuhh.de

Meredith L. Patterson
Upstanding Hackers, Inc.
Brussels, Belgium
mlp@upstandinghackers.com

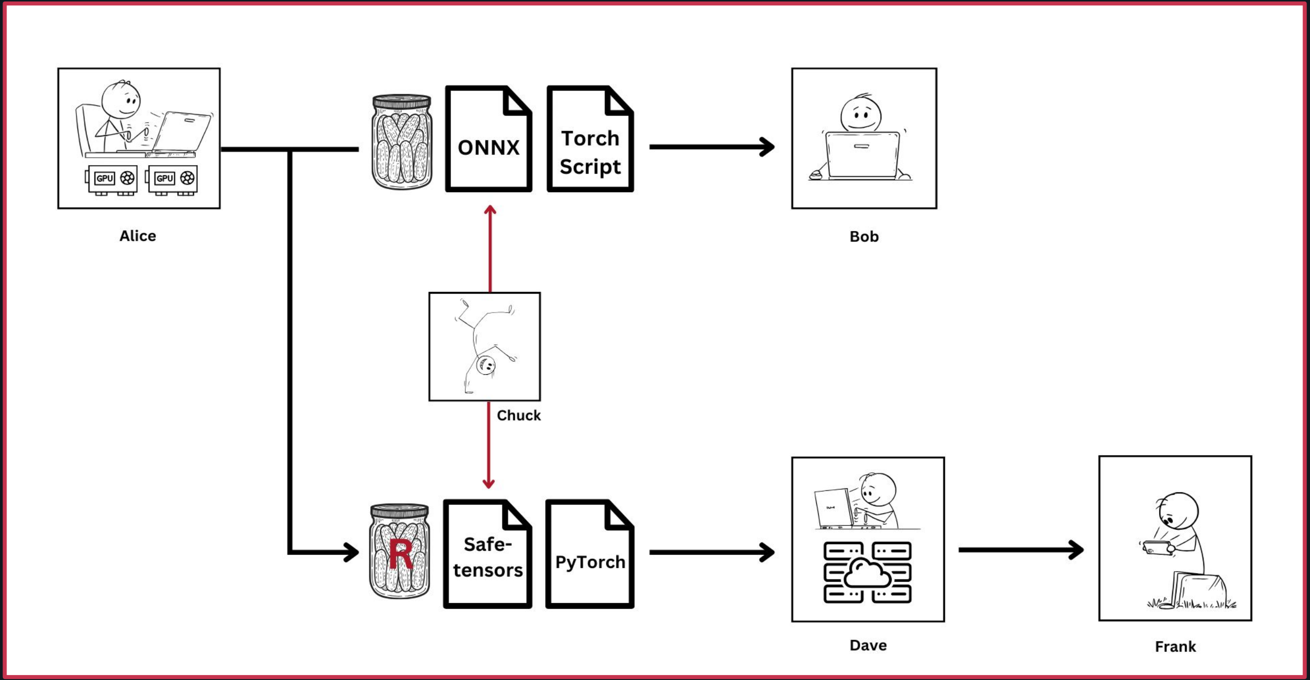
- Non-minimalist input-handling code
- Shotgun parsing
- Input language too complex
- Incomplete protocol specification
- Permissive processing of invalid input
- ~~Overloaded field in input format~~
- Differing interpretations of input language

Secret Formula!

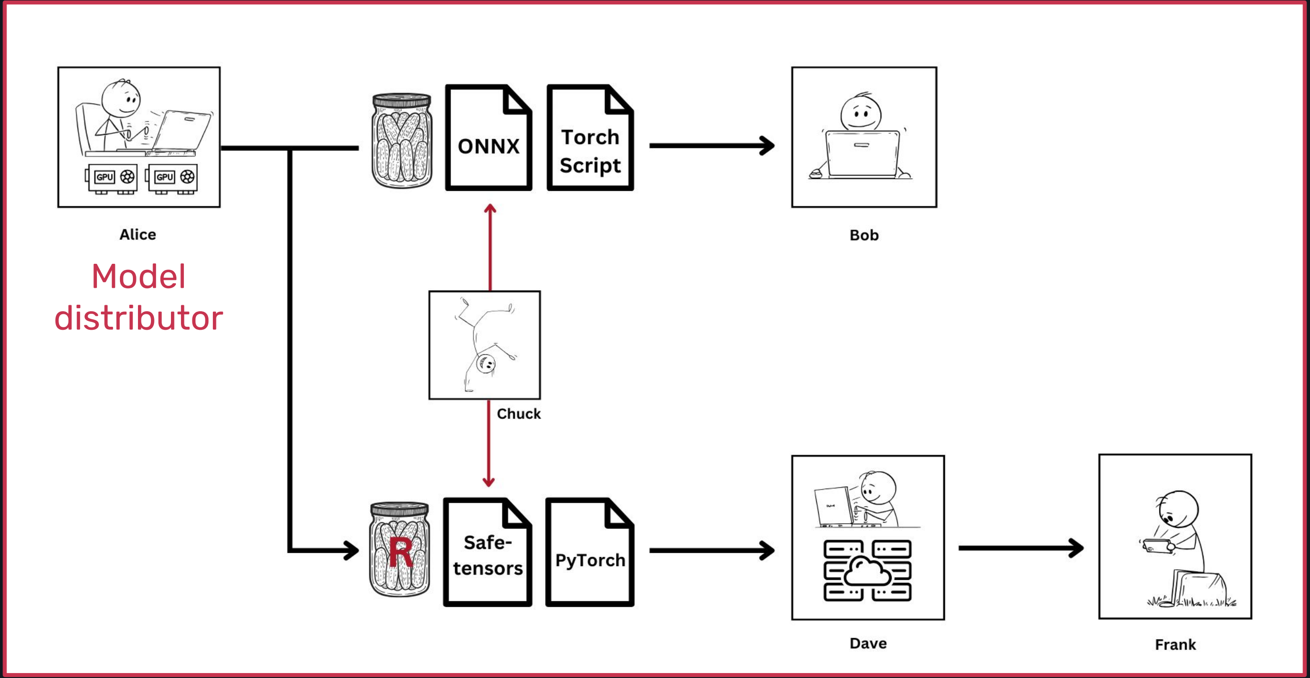
1. Find input-handling bugs with ML model files
2. Categorize these bugs according to the LangSec taxonomy
3. Exploit the bugs to inject a backdoor into ML models
4. ?????
5. Profit?

The Fault in Our Parsers: Incubated ML Exploits

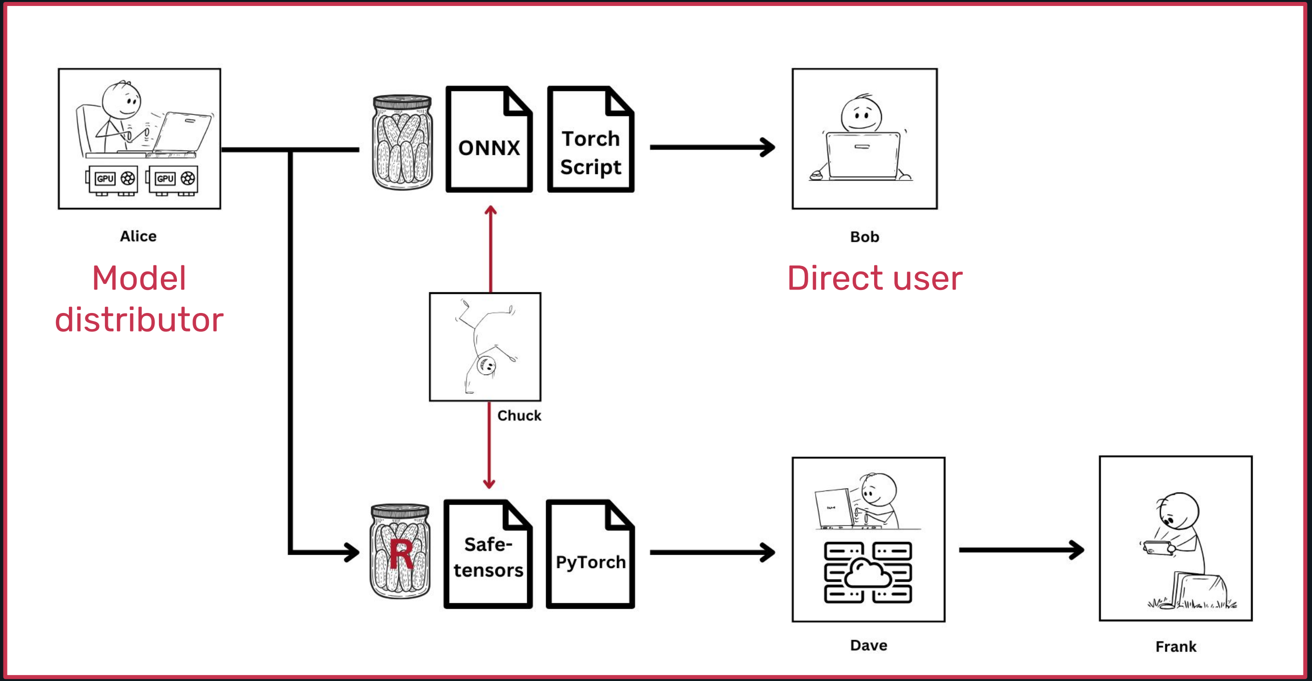
Introducing the cast



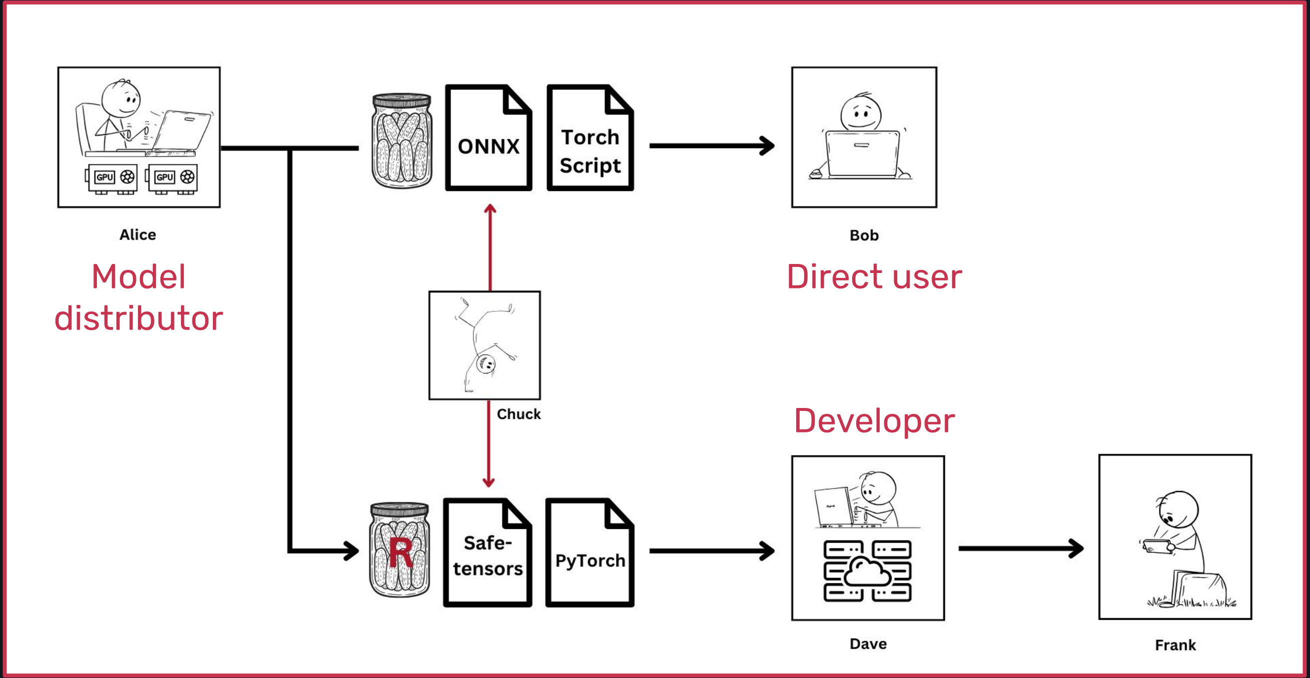
Introducing the cast



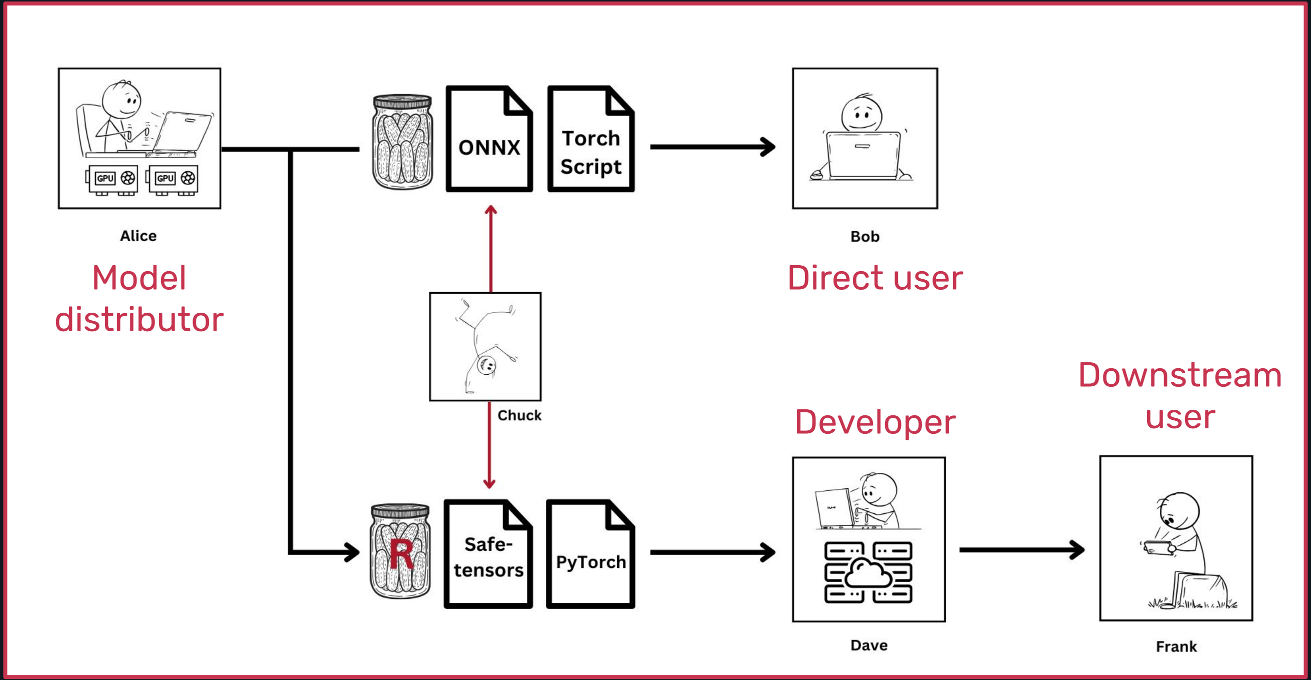
Introducing the cast



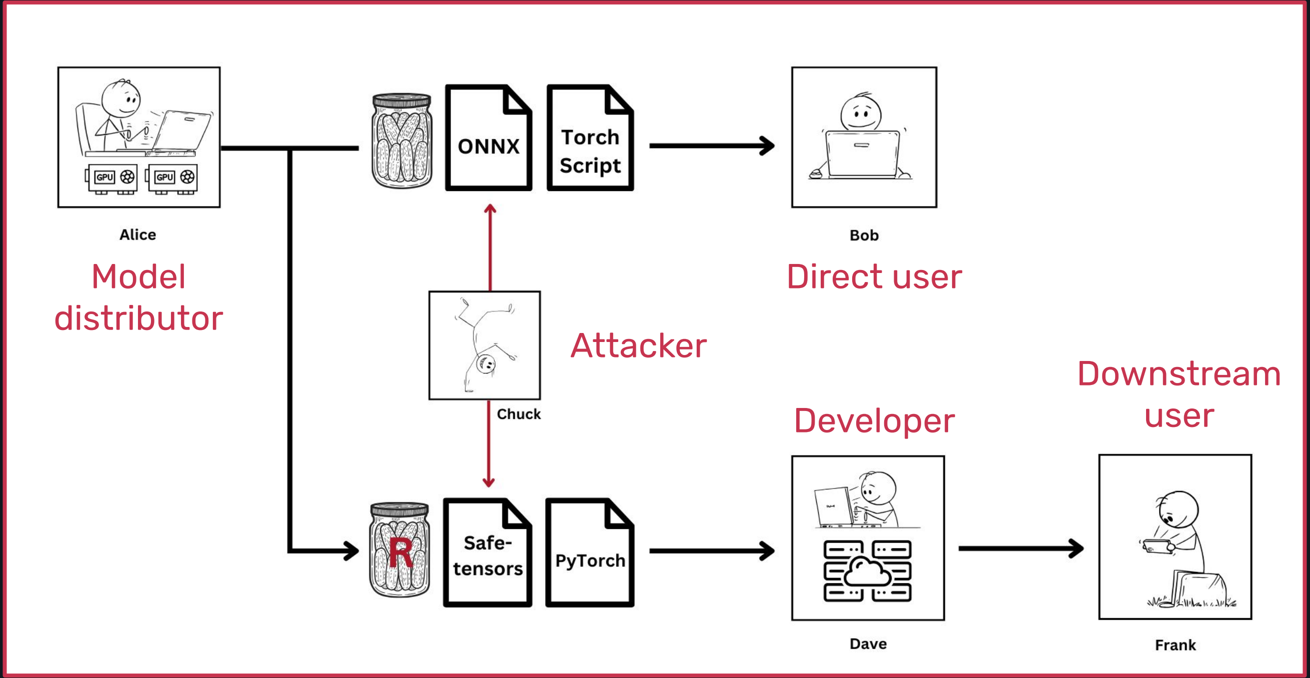
Introducing the cast



Introducing the cast



Introducing the cast



Non-minimalist input-handling code

Exploiting ML models with pickle file attacks: Part 1

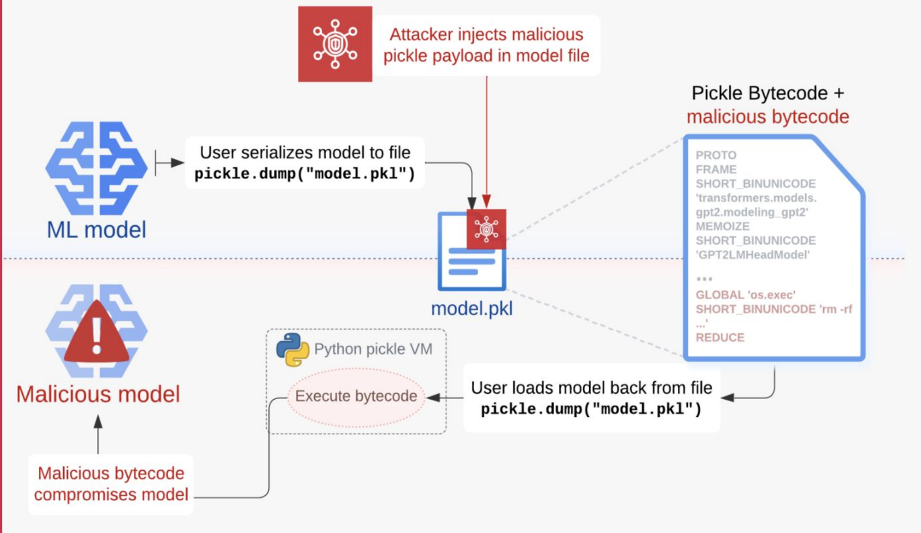


Figure 1: Corrupting an ML model via a pickle file injection

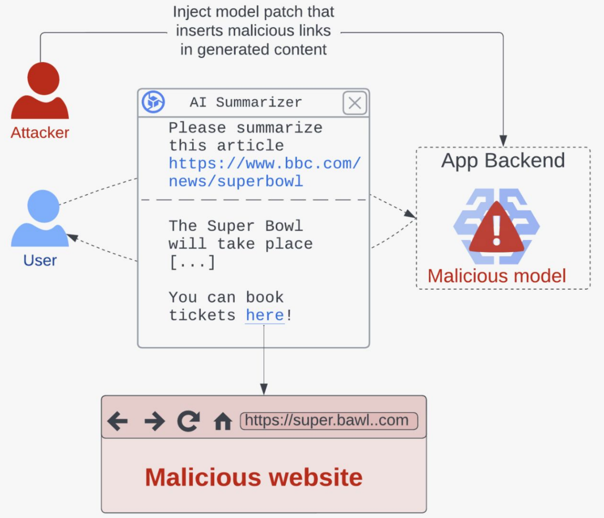


Figure 5: Compromise model to attack users indirectly

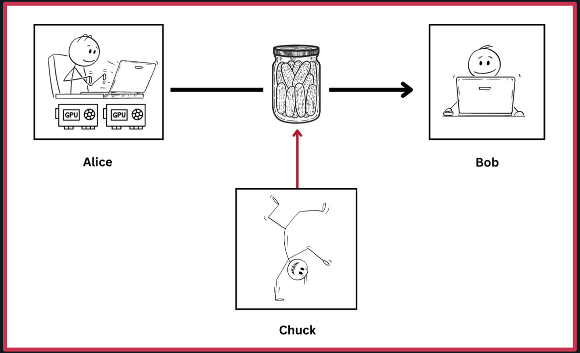
A Pickle for the (Un)Knowing Ones

It's not my fault! It's just GLOBAL and REDUCE!

It's not like I can get surgery...

We can dill with it using Fickling!

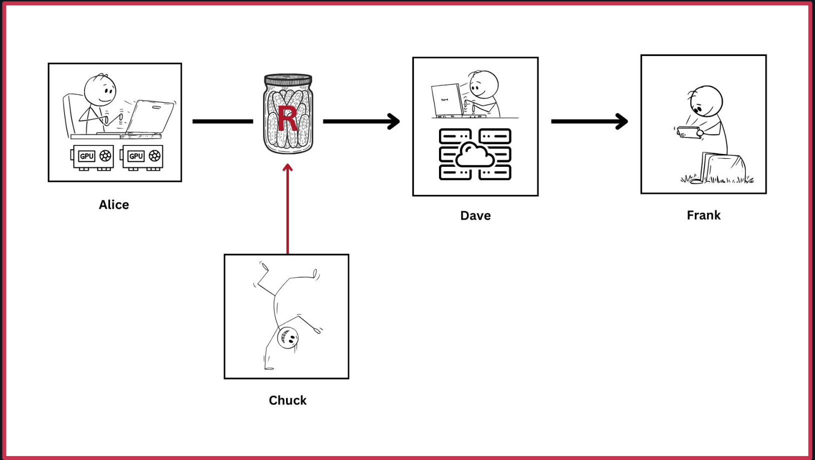
We *relish* the thought of a day when pickling will no longer be used in ML.



- Decompile and statically analyze potentially malicious pickles
  - Pickle VM → Python AST → Human-Readable Python
- Create malicious pickles by rewriting the bytecode
  - Inject arbitrary code
- Analyze and inject code into PyTorch files

Permissive processing of invalid input

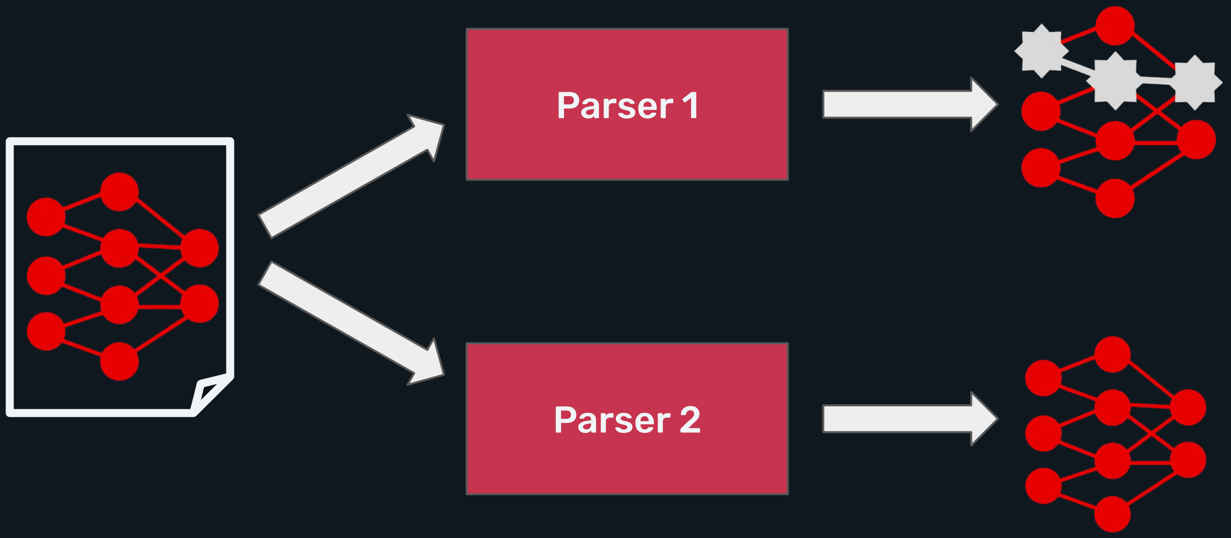
Don't use restricted unpicklers either!



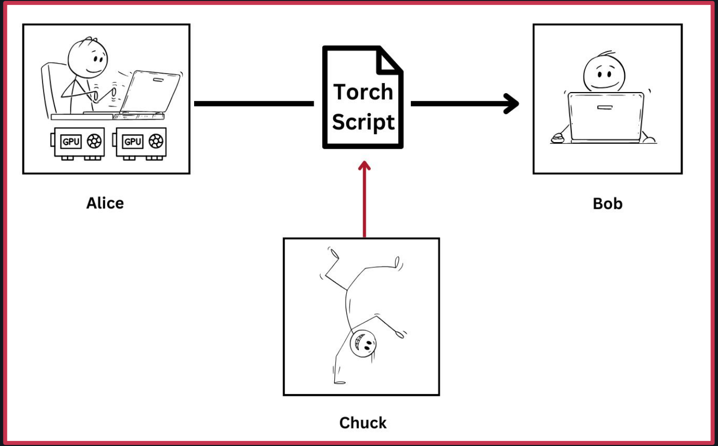
- Restricted unpicklers are a common mitigation
  - Subclass `Unpickler` and override `Unpickler.find_class`
  - Enforces an allow-list or block-list
- Pain Pickle (Huang et al., 2022): 3 general bypass strategies for 8 types of unpickler implementations
- Vast majority of unpicklers can be easily bypassed

Differing interpretations of input language

Parser Differentials



TorchScript differentials



Differential #1

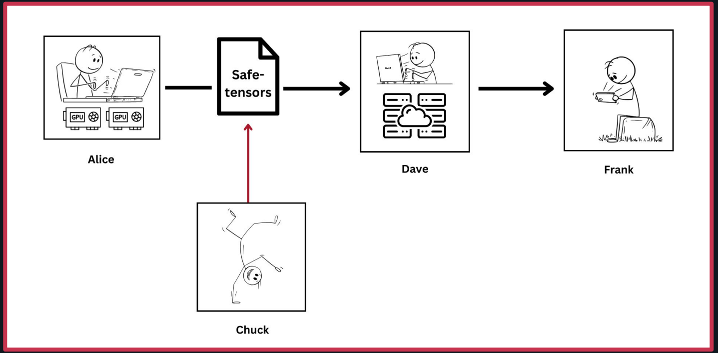
- TorchScript tracing does not incorporate dynamic control flow
- Exploit with architectural backdoor

Differential #2

- Found in the ToB audit of the YOLOv7 codebase
- TOB-YOLO-10: Improper use of TorchScript tracing leads to model differentials
- Constructed an input that caused the two models to behave differently

Safetensors parser differential

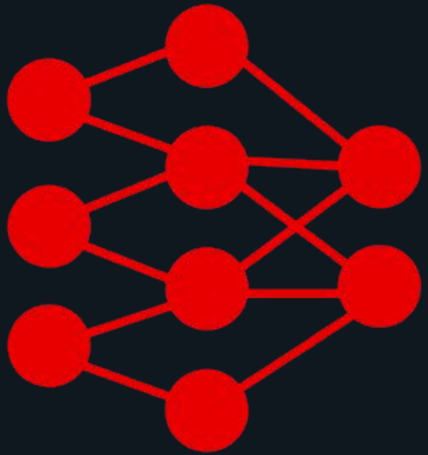
Minimize differentials!



- "Parsing JSON is a Minefield" – Nicolas Seriot
- TOB-SFTN-7: "Underspecified JSON behavior can lead to parser differentials"
- Python built-in JSON parser versus serde JSON parser in safetensors
- Use duplicate key in JSON metadata to force external parsers to load backdoored model weights
    - Your metadata matters!

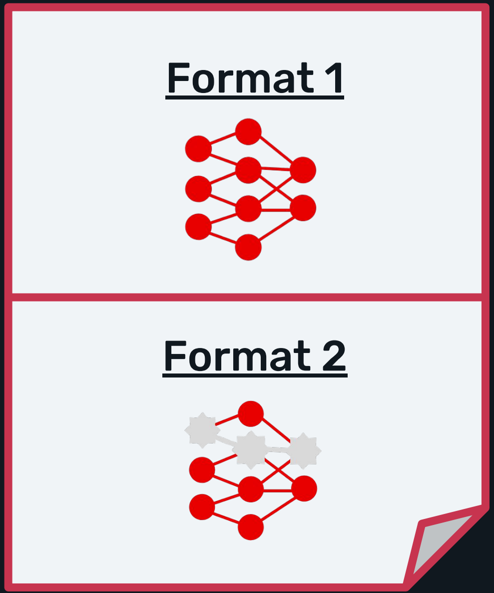
Differing interpretations of input language

- Parser differentials
- Preprocessing differentials -> Change the weights
  - Image-scaling attacks
  - Unicode input exploitation
- Model transformations -> Change the architecture
  - ML compiler backdoors
  - Quantization backdoors
- Model differentials: instances where the same model is interpreted differently



Shotgun parsing

Polyglot Files



Shotgun parsing: Safetensors polyglots

1. Tensor offsets are not checked against the total size of the tensor data	
Severity: **Medium**	Difficulty: **Low**
Type: Data Validation	Finding ID: TOB-SFTN-1
Target: `safetensors/safetensors/src/tensor.rs`	

- Polyglots include: ZIP, PDF, PyTorch model archive, Keras native file format
  - The report itself is a PDF/ZIP polyglot that contains the safetensors polyglots
- TOB-SFTN-1: Failure to check whether the offsets correspond to the tensor size
  - Has been fixed!
- Append arbitrary data to file and/or set the header size to a magic signature

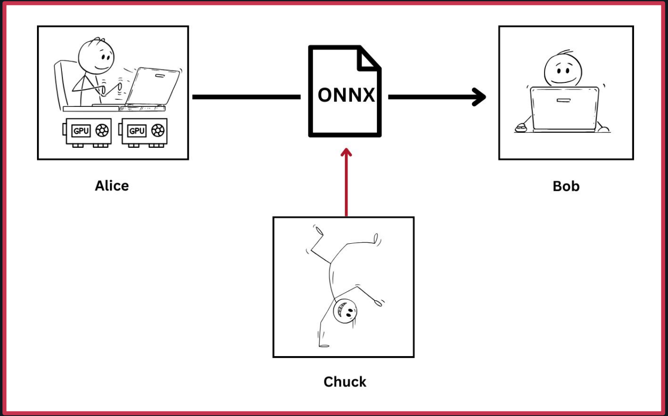
Incomplete protocol specification

Incomplete protocol specification: PyTorch polyglots

- Lacks consistent version numbers
  - PyTorch v1.3/TorchScript v1.4 polymock
- ZIP/Pickle polyglots are easy
  - Pickle allows for appended data
- PyTorch MAR format does not enforce a magic at the start for the ZIP
  - PyTorch v0.1.10 / PyTorch MAR
- Fickling has a polyglot module

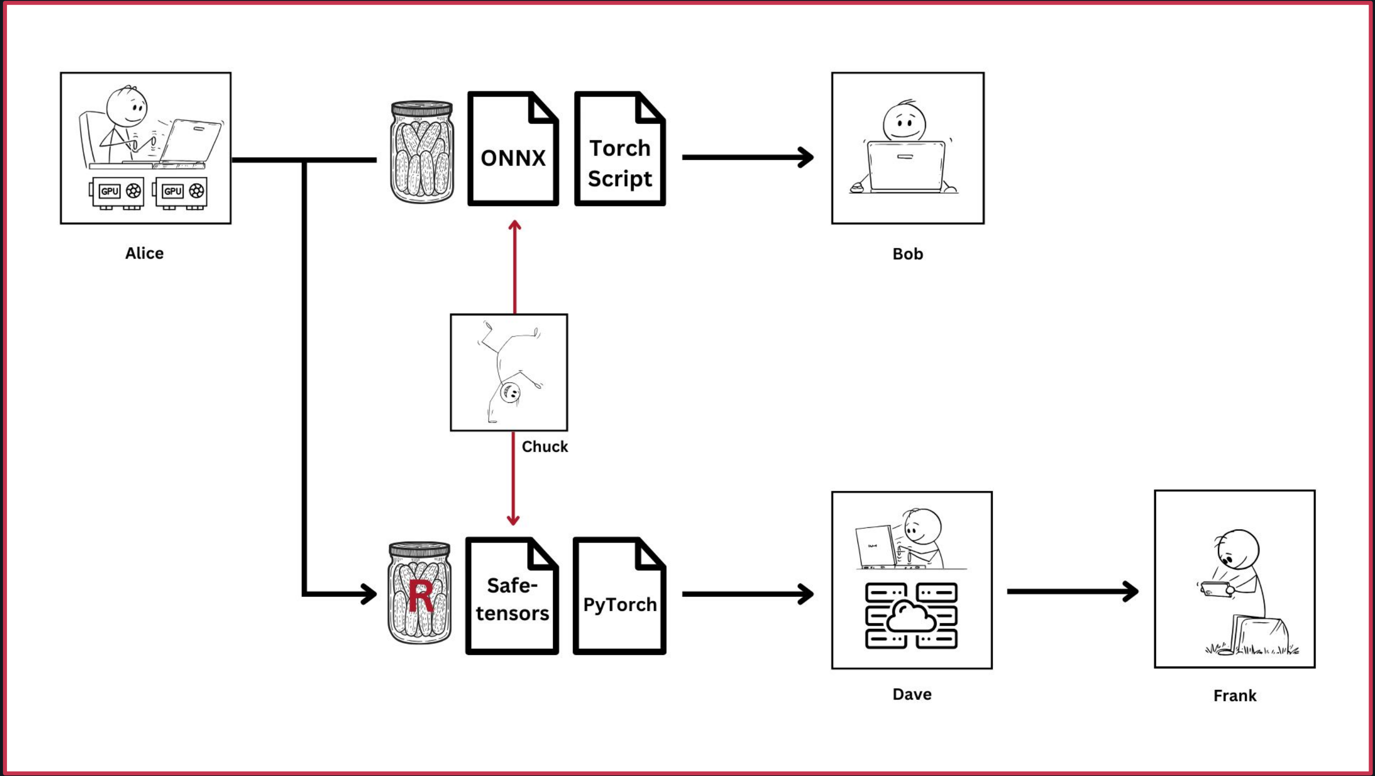
Input language too complex

LobotoMI and the ONNXRuntime



- Problem: A complex input language makes it difficult to determine if a parser is performing correct validation
- Assumption: Specification disallows side effects
- Reality: Arbitrary code can be encapsulated in a custom operation
- Exploit: LobotoMI (https://github.com/alkaet/LobotoMI) + Architectural backdoors

Recap



GPU
GPU
Alice
ONNX
Torch Script
Bob
Chuck
R
Safe-tensors
PyTorch
Dave
Frank

ML Stack Layer	Exploit PoC
Knowledge	
Model	
Framework	• Restricted unpicklers • ONNXRuntime • Pickle
High-Level/Interpreted	
Compiler	• TorchScript differentials
Low-Level	
Infrastructure	• Safetensors polyglot • Safetensors parser differential • PyTorch polyglots
Hardware	

Schema for incubated ML exploits

Core

1. A write primitive to the memory space containing the weights
2. A write primitive to the memory space containing the architecture

Additional

1. Metadata primitives
2. Model transformations & model differentials
3. Custom operators
4. Combine weights and architecture

A brighter future?

1. Model files and metadata are properly validated
   a. Input validation is separate from application logic
   b. Robust and effective trust mechanisms!
2. Differentials and complexity are minimized
   a. No more pickle! No more restricted unpicklers either!
   b. Avoid automata more complex than deterministic context-free
   c. Avoid defining complex input languages
   d. Restrict to ABNF or BNF grammars
3. Better specifications, parsers, protocols, and file formats
   a. File formats should include versions, magic signatures, and checksums
   b. File formats should enforce the signature to be at offset 0
   c. Parsers should reject both prepended and appended data
4. Parameters and architecture are separate

More?

1. Hybrid and incubated ML exploits all over the stack and supply chain
    a. Specific ML tools and contexts
    b. More system security issues
    c. More model vulnerabilities
    d. Look for model differentials
    e. Extend existing PoCs
    f. Frameworks and tools
2. Persistence and reliability of exploits
    a. Mitigations and defenses
3. Secure-by-default software

MLSec Fundamentals

The ML stack and supply chain has not been subject to sufficient security review but is filled with new attack vectors.

The inclusion of ML models into programs changes the system's security properties and introduces new attack surface.

Key Takeaway:

We need to concurrently think about system security issues and model vulnerabilities.

Website: sshussain.me
Twitter/X: @suhackerr
More Info: blog.trailofbits.com

TRAIL OF BITS

EleutherAI, Hugging Face Safetensors Library

Security Assessment

May 3, 2023

Prepared for:
Stella Biderman
EleutherAI

Nicolas Patry
Hugging Face

Garry Jean-Baptiste
Stability AI

Prepared by: **Fredrik Dahlgren, Suha Hussain, Heidy Khlaaf, and Evan Sultanik**

Relishing new Fickling features for securing ML systems

POST MARCH 4, 2024 LEAVE A COMMENT

By Suha S. Hussain

Assessing the security posture of a widely used vision model: YOLOv7

POST NOVEMBER 15, 2023 LEAVE A COMMENT

By Alvin Crighton, Anusha Ghosh, Suha Hussain, Heidy Khlaaf, and Jim Miller

TL;DR: We identified 11 security vulnerabilities in YOLOv7, a popular computer vision framework, that could enable attacks including remote code execution (RCE), denial of service, and model differentials (where an attacker can trigger a model to perform differently in different contexts).

trailofbits / ml-file-formats

References

1. D. Kerr, “Armed with traffic cones, protesters are immobilizing driverless cars,” NPR, Aug. 23, 2023
2. T. Gu, B. Dolan-Gavitt, and S. Garg, ‘Badnets: Identifying vulnerabilities in the machine learning model supply chain’, arXiv preprint arXiv:1708. 06733, 2017.
3. Trail of Bits, ‘Auditing the Ask Astro LLM Q&A app’. [Online]. Available: https://blog.trailofbits.com/2024/07/05/auditing-the-ask-astro-llm-qa-app/.
4. ‘GGUF, the long way around --- vickiboykis.com’. [Online]. Available: https://vickiboykis.com/2024/02/28/gguf-the-long-way-around/.
5. Adia, ‘Data Scientists Targeted by Malicious Hugging Face ML Models with Silent Backdoor --- jfrog.com’. [Online]. Available: https://jfrog.com/blog/data-scientists-targeted-by-malicious-hugging-face-ml-models-with-silent-backdoor/.
6. Gabe, ‘Weaponizing ML Models with Ransomware | HiddenLayer --- hiddenlayer.com’. [Online]. Available: https://hiddenlayer.com/research/weaponizing-machine-learning-models-with-ransomware/.
7. B. Milanov, ‘Exploiting ML models with pickle file attacks: Part 1’. [Online]. Available: https://blog.trailofbits.com/2024/06/11/exploiting-ml-models-with-pickle-file-attacks-part-1/.

References

1. E. Sultanik, ‘Never a dill moment: Exploiting machine learning pickle files’, 2021. [Online]. Available: https://blog.trailofbits.com/2021/03/15/never-a-dill-moment-exploiting-machine-learning-pickle-files/.
2. ‘GitHub - trailofbits/ml-file-formats: List of ML file formats --- github.com’. [Online]. Available: https://github.com/trailofbits/ml-file-formats.
3. ‘GitHub - 0x36/weightBufs: ANE kernel r/w exploit for iOS 15 and macOS 12 --- github.com’. [Online]. Available: https://github.com/0x36/weightBufs.
4. E. Sultanik, ‘EleutherAI, HuggingFace Safetensors Security Assessment’. [Online]. Available: https://github.com/trailofbits/publications/blob/master/reviews/2023-03-eleutherai-huggingface-safetensors-securityreview.pdf.
5. S. F. Jørgensen, ‘llama.ttf --- fuglede.github.io’. [Online]. Available: https://fuglede.github.io/llama.ttf/.
6. ‘LangSec’. [Online]. Available: https://langsec.org/.
7. F. Momot, S. Bratus, S. M. Hallberg, and M. L. Patterson, ‘The seven turrets of babel: A taxonomy of langsec errors and how to expunge them’, in 2016 IEEE Cybersecurity Development (SecDev), 2016, pp. 45–52.

References

1. R. Stevens, O. Suciu, A. Ruef, S. Hong, M. Hicks, and T. Dumitraş, 'Summoning demons: The pursuit of exploitable bugs in machine learning', arXiv preprint arXiv:1701. 04739, 2017.
2. N. Carlini et al., 'Poisoning Web-Scale Training Datasets is Practical', in 2024 IEEE Symposium on Security and Privacy (SP), 2024, pp. 176–176.
3. R. Schuster, J. P. Zhou, T. Eisenhofer, P. Grubbs, and N. Papernot, 'Learned systems security', arXiv preprint arXiv:2212. 10318, 2022.
4. E. Debenedetti et al., 'Privacy side channels in machine learning systems', arXiv preprint arXiv:2309. 05610, 2023.
5. S. Li et al., 'Hidden backdoors in human-centric language models', in Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, 2021, pp. 3123–3140.
6. K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, 'Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection', in Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security, 2023, pp. 79–90.
7. A. Travers et al., 'On the Exploitability of Audio Machine Learning Pipelines to Surreptitious Adversarial Examples', arXiv preprint arXiv:2108. 02010, 2021.

References

1. S. S. Hussain, ‘Relishing new Fickling features for securing ML systems’. [Online]. Available: https://blog.trailofbits.com/2024/03/04/relishing-new-fickling-features-for-securing-ml-systems/.
2. Trail of Bits, ‘Assessing the security posture of a widely used vision model: YOLOv7’. [Online]. Available: https://blog.trailofbits.com/2023/11/15/assessing-the-security-posture-of-a-widely-used-vision-model-yolov7/.
3. E. Quiring and K. Rieck, ‘Backdooring and poisoning neural networks with image-scaling attacks’, in 2020 IEEE Security and Privacy Workshops (SPW), 2020, pp. 41–47.
4. E. Clifford, I. Shumailov, Y. Zhao, R. Anderson, and R. Mullins, ‘ImpNet: Imperceptible and blackbox-undetectable backdoors in compiled neural networks’, in 2024 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML), 2024, pp. 344–357.
5. S. Hong, M.-A. Panaitescu-Liess, Y. Kaya, and T. Dumitras, ‘Qu-anti-zation: Exploiting quantization artifacts for achieving adversarial outcomes’, Advances in Neural Information Processing Systems, vol. 34, pp. 9303–9316, 2021.
6. H. Langford, I. Shumailov, Y. Zhao, R. Mullins, and N. Papernot, ‘Architectural neural backdoors from first principles’, arXiv preprint arXiv:2402. 06957, 2024.