Sai Tanmay Reddy Chakkera
Alfredo Rivero
Shreejay Jahagirdar
Sanskar Sehgal

Malicious Pickles: Keep My Pickles Edible (Literature Review)

Security within deep learning continues to be an area of research within the machine learning community, exhibiting unique, vulnerable attack surfaces and attack types. Xiao et al. provide insights into how these security vulnerabilities are unique, with in-depth descriptions of security risks caused by complex deep learning frameworks such as PyTorch and Tensorflow and their numerous dependencies [9]. These dependencies are of particular interest to Anley et al., who find practical exploits in Tensorflow enabled by Tensorflow's dependency on the serialization module Pickle [10].

It is often necessary to transform a Python object, such as a model's weight dictionary, into a format that enables us to store or transmit the object's data for external use. Pickle provides this functionality in Python, serializing arbitrary Python objects into a portable binary format through the use of the Pickle Virtual Machine (PVM) and assembly-like operations [11]. However, the presence of the PVM, necessitated by the need to serialize even the most complex and arbitrary python object, introduces a few issues. First, speed and space efficiency leave much to be desired in comparison to other serialization tools, gathering condemnation from Ben Frederikson [5] and Ned Batchelder [6]. But, most importantly, Pickle's PVM has become a security sore-spot in recent history. Marco Slaviero's Black Hat 2011 presentation *Sour Pickles* brought notoriety to Pickle as an insecure serialization tool, due primarily to the presence of the GLOBAL and REDUCE opcodes in the PVM [1].

Due to its assembly-like nature, the PVM contains an execution stack that reads operations from pickle files for object reconstruction. Slaviero finds that the GLOBAL opcode can be used to load arbitrary python objects onto the execution stack; when followed by a call to the REDUCE opcode, which calls a stack-bound python object's __reduce__ dunder method for reconstruction, arbitrary code execution can be achieved through this dunder method using different attacks[1][3].

A malicious python author can add their own code into the __setstate__ dunder method which is allowed to run freely during the unpickling process using a REDUCE operation, as identified by Evan Sangaline [2]. Yannic Kilcher demos on ML models however, that if an attacker has no knowledge or access to the victim's unpickling environment they can still mount an eval()/exec() attack encoded as a string in a pickle file that will run silently using REDUCE without specific knowledge of the "unpickler".

Serializer attacks like these are not unique to Pickle, with other serializers including JSON and Node-JS's serialization package also being susceptible to malicious entities. JSON parsers are

able to inject malicious code using specific command arguments if compromised by a third-party [12]. Additionally, Node-JS's serialization module experiences a code injection attack on tampered serialized objects using its unserialize() command[13]. However, neither of these serializers are as robust as Pickle, with JSON and Node-JS's serialization module limited to serializing only primitive data types. Consequently, Pickle and the PVM is a "necessary evil" within Python and the ML community's complex models, leaving Pickle's vulnerabilities intact for the foreseeable future.

However, mitigating attacks found within Pickle, although difficult, is not impossible. Brunson et al. [4] introduce a python library called fickling that detects malicious pickle files. Fickling can accept pickled data streams and decompile them into human-readable Python code that, when executed, will deserialize to the original serialized object. Slaviero et al. [1] introduce Ananpickle, another sanitiser that attempts to secure the PVM. Ananpickle is capable of accepting, examining and manipulating pickle files. Fickling and Ananpickle can be utilized exclusively to detect malicious pickle files and are not sanitization tools. Due to Python being their primary programming language, both tools also suffer from significant performance issues.

We intend on implementing a sanitiser using C++ as the base programming language that can detect and sanitize malicious pickle files. Since C++ is a compiled language, we expect our compiler to be much more efficient. As noted above, both Fickling and Ananpickle can only be utilized to detect malicious pickle files. The sanitizer we wish to implement will go a step further and try to repair malicious pickle files to output a secure version of the file that can be safely deserialized using the PVM. Due to the frequent corruption of Pickle modules when modifications are performed, this is a particularly challenging task.

References:

[1] Marco Slaviero. 2011. Sour Pickles: Shellcoding in Python's serialization format. Retrieved from https://media.blackhat.com/bh-us-11/Slaviero/BH_US_11_Slaviero_Sour_Pickles_WP.pdf

[2] Evan Sangaline. 2017. Dangerous Pickles - Malicious Python Serialization. Retrieved from https://intoli.com/blog/dangerous-pickles/

[3] Nelson Elhage. 2011. Exploiting misuse of Python's "pickle". Retrieved from https://blog.nelhage.com/2011/03/exploiting-pickle/

[4] Evan Sultanik. 2021. Never a dill moment: Exploiting machine learning pickle files. Retrieved from https://securityboulevard.com/2021/03/never-a-dill-moment-exploiting-machine-learning-pickle-files/

[5] Ben Frederikson. 2014. Don't pickle your data. Retrieved from https://www.benfrederickson.com/dont-pickle-your-data/

[6] Ned Batchelder. 2020. Pickle's nine flaws. Retrieved from https://nedbatchelder.com/blog/202006/pickles_nine_flaws.html

[7] Yannic Kilcher. 2022. The hidden dangers of loading open-source AI models. Retrieved from https://www.youtube.com/watch?v=2ethDz9KnLk&t=562s

[8] Jon Boyens. 2012. Notional Supply Chain Risk Management Practices for Federal Information Systems. Retrieved from https://csrc.nist.gov/glossary/term/red_team_blue_team_approach
[9] Qixue Xiao, Kang Li, Deyue Zhang, Weilin Xu. 2017. Security Risks in Deep Learning Implementations. ArXiv Prepint. Retrieved from https://arxiv.org/pdf/1711.11008.pdf
[10]Chris Anley. 2022. Whitepaper – Practical Attacks on Machine Learning Systems. Retrieved from https://research.nccgroup.com/2022/07/06/whitepaper-practical-attacks-on-machine-learning-systems/
[11] Artem Golubin. 2018. How pickle works in Python. Retrieved from https://rushter.com/blog/pickle-serialization-internals/
[12] Trent Mick. 2021. Potential Code Injection Vulnerability in -d Argument. Retrieved from https://github.com/trentm/json/issues/148 JSON arbitrary code exploit,
[13] Ajin Abraham, Bhardwaj Machiraju. 2017. Exploiting Node.js deserialization bug for Remote Code Execution. Retrieved from https://opsecx.com/index.php/2017/02/08/exploiting-node-js-deserialization-bug-for-remote-code-execution/, https://github.com/luin/serialize/issues/4
[14] Priya Pedamkar. Python Pickle vs JSON. Retrieved from https://www.educba.com/python-pickle-vs-json/