Weekly Log

Luke Kurlandski

September 15, 2022

W1: 08/21/2022

Notes

- There are limited examples of applying LMs specifically to malware. Found no examples of using a highly advanced transformer-based architecture, such as BERT.
- Could possibly train a code LM specifically on malware, e.g., MalBERT or MalELECTRA.
- Malware generation using seq2seq? Malware obfuscation using code-repair techniques?
- Several works use word2vec, RNNs, LSTMs for language modeling malware, but there is no one paper that uses all three on the same dataset for a holistic comparison. Furthermore, most of these datasets are not publicly available.
- No paper has used the AST approach taken by Uri Alon et al. specifially for malware
- Perhaps use a large language model of code after decomposition, such as PolyCoder, for the classification? Or would this not work because the decompiled malware would not have useful identifiers? Would it work if ASTs were used to train the LM?
- Is there a BERT for Assembly language? If so, that could be an excellent LM for malware.

Report

- Reading about malware detection and classification
- At this point I'm probably more comfortable/qualified with static malware detection than dynamic detection
- The research in using language modeling techniques on malware is fairly new and undeveloped
- Many papers doing this do not compare their methods to any baseline method
- Many do not use publicly available datasets
- Ideal dataset for this research is probably Sophos-ReversingLabs 20 Million dataset

Minutes

- This week I will work on replicating some of the experimental results produced in other papers

W2: 08/27/22

Onboarding Meeting

- Dr. Pan, Dr. Wright, and I will meet during the Malware Group meeting time, during the reading group when I am presenting, and as needed.
- The RPA is essentially a short research paper with a robust literature review.
- I should begin preparation for it immediately.
  - See rubrics on RIT site.
  - Could focus on a particular type of malware detection, such as Windows macro malware, PDF malware or more general malware detection.
  - Could look into generating adversarial malware examples, along the lines of code2vec.
- "binary lifting" from binary to intermediate (similar to assembly, which could be recompiled again)
- Goal is for second half of semester to have a solid objective for research
- Innovation: if its been done ten times, its not innovative enough
- Read other students' RPAs

Notes

- Adversarial malware: generate embeddings for malware opcode and substitute to fool the detector
- Little research done on poisoning attacks

Report

- Read about adversarial malware generation and defense.
- Most literature is concerned with adapting CV methods that use continuous representations to malware with discrete representations.
- Might be novel to explore adversarial malware with continuous representations, i.e., malware represented using a learned embedding.
- Adversarially modifying the embedding would be simple, but figuring out how that can be achieved by modifying the malware itself would be challenging.
- Look into NLP/CV/audio processing adversarial techniques that use embeddings.
- Dr. Pan suggested and idea of swapping malware API calls with similar calls as an adversarial approach.
- If using embeddings as a feature representation this would fit into that idea nicely.

Minutes

- Develop a plan for the semester

W3: 09/02/2022

Notes

- Would using LM techniques in malware detection make them robust to adversarial attacks?
- Idea: build a [novel?] LM-based classifier, demonstrate its robustness to adversarial attacks, propose a new adversarial attack based upon NLP techniques!!!! Preferably use windows, android, and PDF datasets
- AST for assembly code probably not logical, but would it even be needed?
- Diverse ensembles more robust to adversarial attacks?
- Could be interesting to train an ensemble using dramatically different feature reps, then test an RL, GAN, Graph, and traditional Adversarial malware attacks
- Research on attacks should be focused on the black box problem space scenario.
- Identify problematic sequences of opcode and replace them with computationally equivalent ones to fool detector?
- Adversarial Malware needs to be verified for correctness in Cuckoo Sandbox
- An adversarial attack designed specifically for ensembles?
- Adversarial training works...but is there a point where retraining fails?
- Have RL, genetic, or GAN attacks been tested against ensembles? One paper tested various attacks against a small ensemble, but what about a large one? IE test black box attacks against ensembles, possibly that use different feature reps.
- Test ensembles using different feature rep vs ensembles using the same feature rep
- Read every source that uses black box attacks in the feature space. Determine whether or not their methods use reasonable defense mechanisms. Test their methods using reasonable defense mechanisms.
- Does the surrogate model negate the black box model entirely? Are all black box attacks really white box?
- Testing adversarial techniques against AV software would be more practical than just testing against a single DNN
- Idea: adversarial testing against an ensemble of AV softwares, like VirusTotal

Report

- Read more about adversarial malware evasion attacks
- Observed that many attacks assume unrealistic knowledge about the classification system or are ineffective at or incapable of producing functioning adversarial examples
- Existing black box, practical adversarial evasion attacks generally use “easy” classifiers
- Interesting to see how their techniques hold up against a more realistic defender, like an ensemble or a commercial AV product

Minutes

-

W4: 09/12/22 - 09/19/22

General

- Concerning swapping blocks of suspicious code with less suspicious but equivalent alternatives, the AST approach might be useful because it represents the actual functionality of the code and code produce code that functions the same, but uses weird sequences of tokens.
- Could use code2seq to take assembly to language, then use codex to take language to assembly to get non deterministic outputs
- Interesting paper that uses trains GPT2 on malware bytecode
- Perhaps a code generation tool like codex could repeatedly nondeterministically mask and substitute chunks of assembly for adversarial evasion
- Use codex to modify the source code of malware
- Large scale "instruction substitution" in poisoning attacks?
- Using code generation techniques to change the important code itself is an extremely challenging problem. Several methods use various strategies to append some form of bytes into sections of the code. We could use code generation techniques to insert logically functional code blocks into the malware. At that point though, we might as well just copy and paste assembly from a specific source.
- Could use code generation to write C code, then compile it, and insert the compiled code into the malware. Use genetic or RL algorithms to find where to insert it, or simply insert it behind a if-false statement.
- Code summarization of a portion of assembly code via code2vec or something like it. Then code generation from the summary back into assembly.
- To create a labeled assembly language, compile C code and use function names as labels.
- Shellcode_IA32 has some code generation with codeBERT
- PalmTree is a really effective embedding setup

Malware Group

Report

- Interested in Adversarial Malware Evasion Attacks

Minutes

- Code generation techniques to produce adversarial examples, eg, code2vec/code2seq on assembly
- Would have to generate code that functions, but doesn't appear malicious.
- Oakland paper Saidur shared in the Slack group
- Android research might be more promising than PE research
- Benign Android data is easier to attain
- AndroidZoo is a widely used dataset
- Dr. Pan will send out some papers

Reading Group

-

Weekly Scrum

Report

- Looking into using LLMs for adversarial malware generation
- Hard problem and not exactly sure what directions to focus on
- The start would be with a language model at the assembly or binary level
- The state-of-the-art seems to be PalmTree (Pre-trained Assembly Language Model for InsTRuction EmbEdding), which uses a BERT architecture with different training tasks adopted for code
- This gives us really good embeddings for assembly instructions
- BERT has been adopted for sequence tasks in natural language and in code tooling, so there could be some extensions made to PalmTree to get interesting things going at the assembly level
- Took a little bit of wandering to find PalmTree and other binary analysis tools
- This weekend, will develop more concrete ideas and write up that abstract

Minutes

-

W5: 09/12/22 - 09/19/22

General

-

Malware Group

Report

-

Minutes

-

Reading Group

-

Weekly Scrum

Report

-

Minutes

-

W6: 09/12/22 - 09/19/22

General

-

Malware Group

Report

-

Minutes

-

Reading Group

-

Weekly Scrum

Report

-

Minutes

-

WN: XX/XX/XX - XX/XX/XX

General

-

Malware Group

Report

-

Minutes

-

Reading Group

-

Weekly Scrum

Report

-

Minutes

-