Generative Adversarial Networks in Cybersecurity: An Evolution of Malware Detection

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Trevor Williams

Spring, 2022

Daniel Graham, Department of Computer Science

Generative Adversarial Networks in Cybersecurity: An Evolution of Malware Detection

CS4991 Capstone Report, 2022

Trevor Williams
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
Tew4fs@virginia.edu

Abstract
Malware detection is a field that needs to constantly evolve with the improvement of malware concealment. The use of generative adversarial networks (GANs) could be the next step in that evolution. GANs allow for two agents to learn opposing traits by competing against each other. One agent produces malware for the other agent to detect. Ideally, one agent will eventually be able to distinguish between benign and infected programs. More research is still needed to evaluate if and how GANs can be utilized in fighting malware.

1 Introduction

The COVID-19 pandemic has turned the world's focus towards technology and the internet more so than ever. With more time being spent online, there have been a correlated increase in cybercrime. Kaspersky, an antivirus software company reported finding an average of 380,000 infected files daily in 2021. An increase of about 20,000 files from 2020 [4]. With more attention being brought to malware development, the quicker antimalware systems must be in adapting. Most notably, the better they must be at detecting zero-day exploits.

A zero-day exploit is how a cybercriminal attacks a vulnerability in a system before the system's owners become aware of the vulnerability. These exploits are difficult to spot by anti-malware software as these are completely novel exploits. Machine learning has been a popular approach to combat this because it allows for anti-malware to adapt to new situations. Bitdefender, a popular antivirus software has shown success using machine learning. In a 2020 test suite, Bitdefender was the only software to detect all 15 advance persistent threats [1]. However, malware continues to evolve, and anti-malware techniques must as well. One potential route is the use of GANs.

2 Background

Malware detection has come a long way from its early foundation. This foundation consisted of mainly using signature-based detection to identify an infected program. Signature-based detection stems from the fact that early viruses always had a specific sequence of instructions that the anti-malware could use to detect that virus. Metamorphic viruses implement a way for the virus to mutate between its hosts. This mutation keeps the virus intact but changes the sequence of instructions for better concealment.

Since signature-based detection is no longer viable for these types of viruses, an array of ideas has been proposed to fight metamorphic viruses. One idea is code emulation in which an anti-malware software will execute a program in a sandbox environment to examine if any malicious results occur. Of course, malware writers adapted to detect if the program was being run in a sandbox environment to try and trick the anti-malware systems. Overall, this co-evolution has been an ongoing fight that resembles the training process of GANs.

GANs consist of two models training simultaneously. One model generates sample data for the other model to discriminate against [2]. The discriminator model estimates whether a sample came from real data or was generated by the other model. Through iterations each model evolves in efforts to beat out the other model. An application of GANs is image classification. The generator model creates images that the discriminator model estimates are real or not. Eventually, the generator model can construct images that are indistinguishable from real images.

3 Related Works

The idea of GANs in cybersecurity is not new but it is relatively unexplored. Amin et al. (2022) explored GANs in Android malware detection. More specifically, they looked at Deep Convolution GANs which take advantage of using convolution rather than neural network matrix multiplication. Through their implementation, they concluded that GANs were effective at malware detection and zero-day malware detection. While promising, their work was only limited to Android smartphones which leaves questions as to whether the same conclusions could be made with Windows or Linux machines.

Kim et al. (2018) looked at GANs for specifically identifying zero-day malware. The idea was to use the GANs to generate fake malware samples and distinguish between those and real samples. The model achieved a 95.74% average classification accuracy and was effective against zero-day malware.

Tam and Truong-Huu (2022) used GANs to just generate samples for a baseline classifier. Their method included generated malicious and benign samples to increase the quality of their training set. This proved to be an effective approach as without the generated samples, the model's accuracy dropped more than 10%. Clearly, the use of GANs for malware detection has various approaches to create meaningful results.

4 Proposed Solution

The proposed solution is similar to the research of Kim, et al. However, the difference is that real malware samples will be generated and ~~that~~ the discriminator will be differentiating between benign and malicious programs.

The generator mimics the mutation process that viruses go through. Starting from known viruses, the generator will create new versions, which will look unfamiliar at the code level but could act as a real virus. It is important that the generated versions are as real as possible so that any potential patterns in the architecture of code of the viruses is present.

The downside is that the generator model, through training, could eventually be able to create hard-to-detect viruses. This means that such a model could be used for the intent of creating viruses. One potential fix is to have the generator model omit certain parts of the virus such as the jump to the payload. However, a fix like this could be removing code that could help the discriminator distinguish between a virus and benign program.

The job of the discriminator is to differentiate between malicious and benign programs. The malicious files will come from the generator while the benign file will come from a dataset. It is possible to use another generator here with the sole purpose of creating benign programs. This generator would not have to do any learning because it is only supplying test data to use, rather than trying to trick the discriminator.

Therefore, the overall system would work and train as follows: The discriminator would be presented with a file. After examining the file, it would determine whether it contains a virus or not. If the file was deemed to be malicious and it did come from the generator model, the discriminator was right. Therefore, the discriminator would update its neural network to reinforce that it was right, and the generator would update its neural network to adjust to being wrong. The opposite would happen if the discriminator inaccurately classified a malicious file as benign. The file presented to the discriminator can also come from a test set instead of the generator. In this case, the discriminator is the only model that updates depending on whether it was right or wrong.

5 Expected Results

After many iterations of training, there would be two models. One model would generate viruses with the intent of obfuscation. This model should be discarded as it contains the possibility for malicious use.

The more useful model for virus detection is the discriminator model that can effectively differentiate between malicious and benign files. This model can then be applied to actual files. After the training process is completed, if a file is supplied to the model, it should be able to classify it rather quickly. This means this model could be incorporated into anti-malware software and used on most computers.

Ideally, the model will have an accuracy in the upper 90 percent while limiting both false negatives and false positives. Since the model was trained with no prior knowledge of viruses, it should be able to draw its own unique conclusions about what makes up a virus. This will allow for detection without the use of signatures and detection on novel viruses.

The combination of all these things will bring a better user experience for scanning for malware. With the integration of the model into anti malware software, the user will be able to run scans on their own computer with a quick and effective way of detecting zero-day viruses and metamorphic viruses. All in all, this would create a safer environment for users.

6 Conclusion

The presence of technology in society is only increasing. With that, the use of malicious software will also grow. In response, anti-malware software must continue to get better. Generative adversarial networks provide a foundation to create systems that are effective at detecting malicious viruses. Through the training process, the discriminator will learn how to distinguish between benign programs and those affected by never-before-seen viruses. Therefore, the model can be applied to real software and be able to detect malicious software before damage can be done.

7 Future Work

There is still much work to be done to see a full implementation of this idea. For instance, there needs to be further research as to what type of neural networks should be used. This would depend on the route of implementation and how the training files are being represented for the network to use. Likewise, the number of layers that a neural network has will drastically affect performance. So, testing at different levels would need to be done to address that.

Along with that, an actual dataset of benign files would need to be collected for the training files. Work would also need to be done to ensure that the generator model is creating malicious files. Therefore, a criterion would need to be developed so that the generator knows what needs to be done to create a meaningful malicious file. Perhaps research here can be inspired by how current metamorphic viruses mutate between their hosts.

If a full implementation is reached, testing and evaluation with real world data would need to be done to compare against the current state of the art. Regardless of whether the implementation performs better or worse, hopefully it will provide insight into how to move forward in the development of anti-malware software.

References

[1] AV Comparatives. 2020. Enhanced Real-World Test 2020. Retrieved from https://www.av-comparatives.org/tests/enhanced-real-world-test-2020-enterprise/

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. 2014. Generative Adversarial Nets. *Advances in neural information processing systems*. Vo. 27.

[3] Jin-Young Kim, Seok-Jun Bu, and Sung-Bae Cho. 2018. Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders, Vols. 460-461. https://doi.org/10.1016/j.ins.2018.04.092

[4] Kaspersky. 2021. New malicious files discovered daily grew by 5.7% in 2021. Retrieved from https://usa.kaspersky.com/about/press-releases/2021_new-malicious-files-discovered-daily-grew-by-57-in-2021

[5] Muhammad Amin, Barbar Shah, Aizaz Sharif, Tamleek Ali, Ki-Il Kim, Sajid Anwar. 2022. Android malware detection through generative adversarial networks. *Transactions on Emerging Telecommunications Technologies*. Vol. 33. https://doi.org/10.1002/ett.3675

[6] Wee Ling Tan and Tram Truong-Huu. 2020. Enhancing Robustness of Malware Detection using Synthetically-adversarial Samples. *IEEE Global Communications Conference*, December 7-11, 2020, ELECTR NETWORK. https://doi.org/10.1109/GLOBECOM42002.2020.9322377