Mirai Detection using Generative Adversarial Networks

By: Gavin Wong [gavimwong@gmail.com]

Abstract

Cybersecurity has been a dire issue in our lives as the advancement and usage of technology gradually increases. In the pandemic where many people are performing many tasks online, data is being stored and retrieved at an even more rapid pace, which means increased vulnerability for cyber terrorists to implant malware to access private data. A particular type of malware, Mirai, transforms networked devices into remotely controlled bots through continuous scanning of the internet for the IP address of Internet of Things (IoT) [2]. In this short paper, we attempt to analyze the utilization of a generative adversarial network (GAN) in detecting Mirai intrusion using data inputs through Mirai packet capture (PCAP) files. We will discuss the data cleaning and feature engineering as well as the implementation of a software analysis framework (Zeek) to convert PCAP files into connection logs. We will present the framework for the GAN model and introduce the model components as well as the influence from previous GAN literature. Finally, we will inspect the evaluations and learn the capabilities and benefits of using GAN in a cyber-intrusion detection environment.

Introduction

Mirai malware is a malicious program that locates, attacks and infects vulnerable IoT devices at a swift pace. It builds a botnet through controlling infected devices via a central set of command and control (C&C) servers. These servers dictate the infected devices which sites to attack next. Mirai is made of two key components: a replication module and an attack module [3].

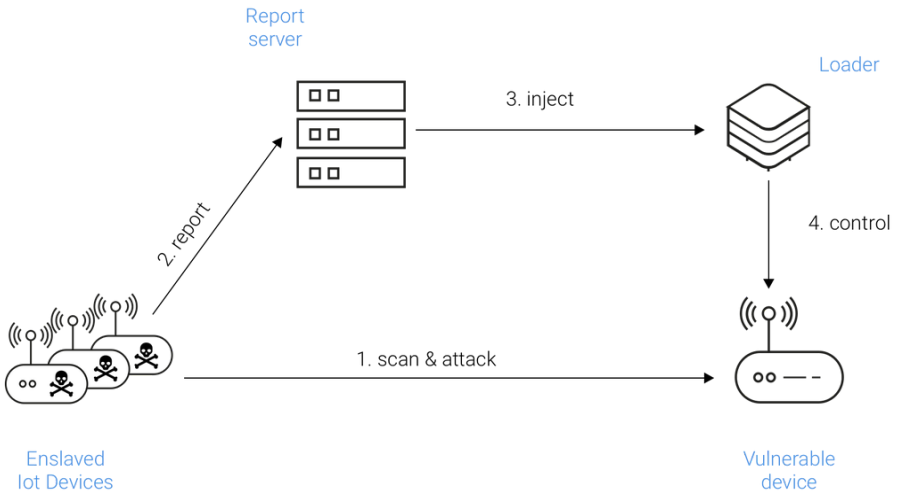

Figure 1. *A diagram showing how Mirai replicates across IoT*

As shown in Figure 1, the replication module is responsible for growing the botnet size by enslaving as many vulnerable IoT devices as possible. It accomplishes this by randomly scanning the entire Internet for viable targets and proliferates by haphazardly sending TCP SYN probes to pseudo-random IPv4 addresses, on Telnet TCP ports 23 and 2323 [1]. Once it compromises a vulnerable device, the module reports it to the C&C servers so it can be infected with the latest Mirai payload. To compromise devices, the initial version of Mirai relied exclusively on a fixed set of 64 well-known default login/password combinations commonly used by IoT devices. While this attack was very low tech, it proved extremely effective and led to the compromise of over 600,000 devices [3].

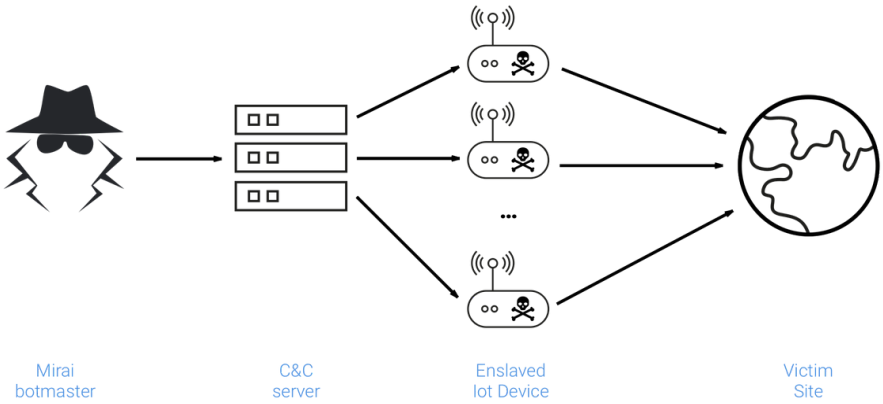

Figure 2. *A diagram showing how Mirai DDoS attacks occur*

Looking at Figure 2, the attack module is responsible for carrying out DDoS attacks against the targets specified by the C&C servers. The Mirai botmaster tells each node in the botnet to launch an attack with specific details such as instructions for HTTP flooding, UDP flooding, and all TCP flooding options. This wide range of methods allows Mirai to perform volumetric attacks, application-layer attacks, and TCP state-exhaustion attacks as node immediately executes the desired attack, sending packets as quickly as possible with no rate limit [5].

The Mirai malware is most infamous for the attacks in 2016: an attack on computer security journalist Brian Krebs' website, an attack on French web host OVH, and the Dyn cyberattack. The most notable attack being, Mirai DDOS attack against Dyn, a major dynamic Domain Name Service (DNS) provider, which was assaulted by a one terabit per second traffic flood and becomes the new record for a DDoS attack [6]. The traffic tsunami knocked Dyn’s services offline, rendering a number of high-profile websites including GitHub, HBO, Twitter, Reddit, PayPal, Netflix, and Airbnb, inaccessible [3].

Now that we understand more context about Mirai, we will continue to explore the datasets implemented to obtain such powerful malware

Dataset

For our datasets, we utilize Zeek (formerly called Bro), an open-source software network analysis framework, within a Ubuntu Virtual Machine environment to generate different log files from packet capture files (PCAP), such as connection.log, dns.log, http.log, and ftp.log. For this particular model, we mainly focused on studying connection logs (conn.log). To understand the type of data and information we extracted, refer to the table (Figure 3) below.

conn.log

IP, TCP, UDP and ICMP connection details

Field	Type	Description
ts	time	Timestamp
uid	string	Unique ID of Connection
id.orig_h	addr	Originating endpoint's IP address (AKA ORIG)
id.orig_p	port	Originating endpoint's TCP/UDP port (or ICMP code)
id.resp_h	addr	Responding endpoint's IP address (AKA RESP)
id.resp_p	port	Responding endpoint's TCP/UDP port (or ICMP code)
proto	transport_proto	Transport layer protocol of connection
service	string	Dynamically detected application protocol, if any
duration	interval	Time of last packet seen – time of first packet seen
orig_bytes	count	Originator payload bytes; from sequence numbers if TCP
resp_bytes	count	Responder payload bytes; from sequence numbers if TCP
conn_state	string	Connection state (see conn.log:conn_state table)
local_orig	bool	If conn originated locally T; if remotely F. If Site::local_nets empty, always unset.
missed_bytes	count	Number of missing bytes in content gaps
history	string	Connection state history (see conn.log:history table)
orig_pkts	count	Number of ORIG packets
orig_ip_bytes	count	Number of ORIG IP bytes (via IP total_length header field)
resp_pkts	count	Number of RESP packets
resp_ip_bytes	count	Number of RESP IP bytes (via IP total_length header field)
tunnel_parents	set	If tunneled, connection UID of encapsulating parent (s)
orig_cc	string	ORIG GeoIP Country Code
resp_cc	string	RESP GeoIP Country Code

Figure 3. *Table depicting network connection details inside Connection Log [7]*

For our benign data samples, we incorporated datasets built by CTU University in Czech Republic, which provides dozens of captures consisting of gigabytes of network traffic data. After evaluating different captures or scenarios, we discovered the CTU-Normal-32 dataset, which contains 66,890 data points in the conn.log, to be the most appropriate and reliable dataset [8].

For our Mirai malware data samples, we incorporated Institute of Electrical and Electronics Engineers (IEEE) IoT Network Intrusion Dataset [9]. Finding an appropriate and reliable Mirai malware dataset proved to quite arduous as many datasets we tested, such as the Kitsune Network Attack Dataset from UCI Machine Learning Repository [9] and CTU-IoT-Malware-Captures from Aposemat IoT-23 Dataset [10], contained many invalid and missing data within their PCAP files. After amalgamating

multiple conn.log from all of the PCAP files collected, we arrive at Mirai dataset containing 39,503 data points. Despite the apparent disparity between the datasets for our two classes, the difficulty in locating more suitable dataset hinders us from fully balancing the dataset. Thus, we maintain our current dataset and continue to feature engineering and selection.

For our final dataset, we decided to only consider these features: id.orig_h, id.orig_p, id.resp_h, id.resp_p, proto, duration, orig_bytes, orig_pkts, orig_ip_bytes. The other features were omitted as not enough information could be extracted to perform sufficient imputation or the features were considered to be irrelevant, such as ts (timestamp). We applied label encoding to the features containing strings or transport protocols [proto], applied feature scaling to the remaining variables. [id.orig_p, id.resp_p, duration, orig_bytes, orig_pkts, orig_ip_bytes], finally, one-hot encoding to the features containing Internet Protocol (IP) address data type [id.orig_h, id.resp_h]; however, due to tremendous variety and inconsistency, we labeled all IP addresses that appeared less than 5 times under an "other" category instead of its own to compact our input data.

GAN Model

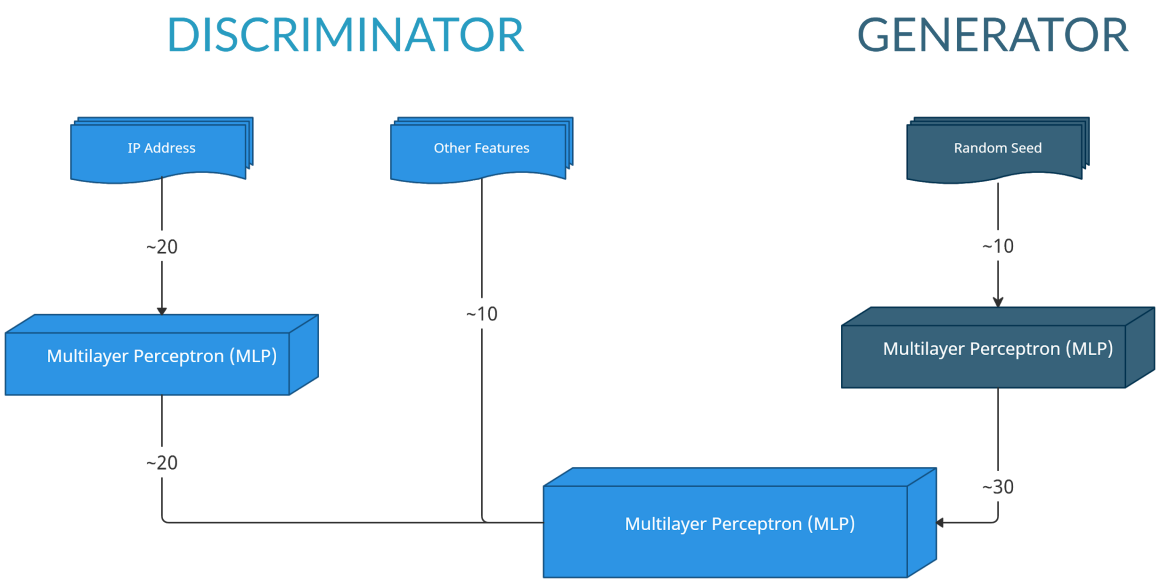

Figure 5. *A diagram for the GAN Model structure implemented*

The Figure 5 above summarizes our GAN Model design and structure. We incorporated additional layers to reduce the size of the input vector from the IP features, essentially reducing the input size from 1224 dimension data to exactly 30. If no reduction was made, it would hinder the training of our model training significantly as well as permit an immediate overfitting. Furthermore, we took a slight Semi-Supervised GAN approach in learning whether our discriminator model is able to distinguish between benign and Mirai network traffic as well as whether it can identify if a data sample seems to be fake and generated by our generator.

Our GAN was able to distinguish between benign and Mirai network traffic, achieving a near 100% validation accuracy and 0 validation loss. These results were expected as the simplicity of merely identifying Mirai malware would allow a regular Multilayer Perceptron (MLP) to achieve the same results. In comparing real data to generated data, our discriminator performed proficient as well. Prior to our GAN training, we performed moderate pre-training to familiarize the discriminator with real data. Afterwards, we trained the GAN through batches of 200 samples or half-batches of 100 samples, where each training cycle is labeled as a training "step." To train for 10 epochs, we calculated that to be 1,500 training steps for our particular dataset. After each step, we evaluated the discriminator through plotting the model's accuracies with a validation test set consisting of real data and generated fake data from the generator using Matplotlib PyPlot. The accuracy values are displayed below along with the corresponding training step.

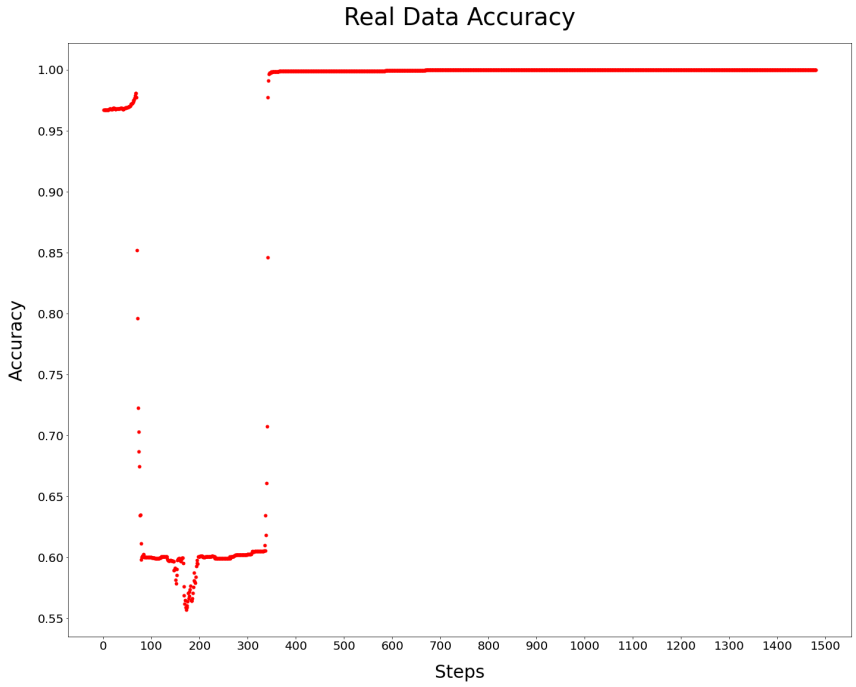

Figure 6. *Plot of Discriminator Accuracy with Real Test Data*

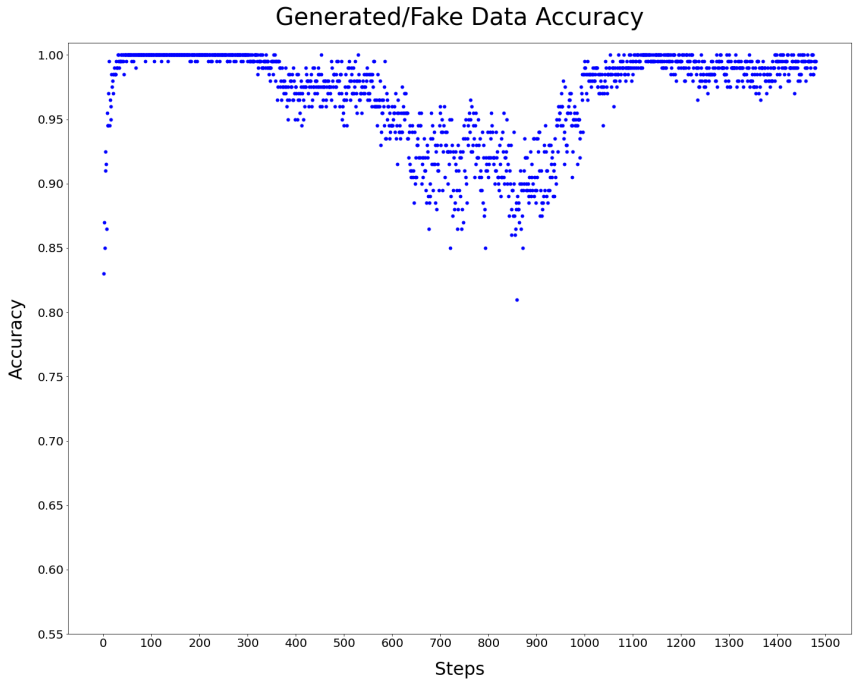

Figure 7. *Plot of Discriminator Accuracy with Fake Test Data*

Looking at Figure 6 and 7, we notice that the pretraining allows the discriminator to have an adequate initial accuracy with classifying real data and a mediocre initial accuracy with

classifying generated data. A drop in accuracy for the real data occurs around step 80, which is most likely due to the consequence of the discriminator beginning to place more weight or emphasis on learning from the generated data and less weight on the real data as the loss values for classifying the generated data much greater. We can also notice the accuracy for generated data rises around this same training time. As we continue our training, we discover that our real data accuracy increases towards nearly 100% in ~500 steps; meanwhile, our generated data accuracy slightly diminishes before also achieving roughly 100% with certain variability after ~1100 steps. Overall, it seems that our discriminator accomplished notable results in analyzing and classifying between benign and Mirai network traffic and real and generated data.

Conclusion

We have demonstrated the usage of the GAN model to detect Mirai as well as distinguish between real and generated data is certainly practicable and achievable, as proven through our metric evaluation and results. Though, GANs are usually implemented for situations and problems involving much higher complexity. In this situation, the use of a regular Multilayer Perceptron (MLP) would achieve similar results in the event of merely classifying Mirai malware. At the moment, we are solely endeavoring to explore the implementation of a GAN-based model in a cyber-intrusion setting, and its latent benefit of being able to generate data. In the future, we should look to explore higher advanced GAN-based models in detecting and dealing with more complicated cyber anomalies.

References

[1] Security, A. (2019, January 30). Mirai botnet explained - history: STRUCTURE: ATTACKS. Retrieved January 28, 2021, from https://blog.attify.com/how-mirai-botnet-hijacks-your-devices/

[2] Blaine, G. (2020, December 8). Inside the Mirai Malware That Powers IoT Botnets. A10 Networks. https://www.a10networks.com/blog/inside-the-mirai-malware-that-powers-iot-botnets/

[3] Author, G. (2020b, August 21). Inside the infamous Mirai IoT Botnet: A Retrospective Analysis. The Cloudflare Blog. https://blog.cloudflare.com/inside-mirai-the-infamous-iot-botnet-a-retrospective-analysis/

[4] Dulauno, A., Wagener, G., Wagner, C., & Mokaddem, S. (n.d.). AN EXTENDED ANALYSIS OF AN IOT MALWARE FROM A BLACKHOLE NETWORK [Web log post]. https://www.circl.lu/assets/files/tnc17_paper_Fullpaper-IoTBlackholeCW.pdf

[5] Shoemaker, G. A. A. (2017, April 10). How to Identify a Mirai-Style DDoS Attack. Blog. https://www.imperva.com/blog/how-to-identify-a-mirai-style-ddos-attack/

[6] Five most Famous ddos attacks and then some. (2020, December 10). Retrieved February 03, 2021, from https://www.a10networks.com/blog/5-most-famous-ddos-attacks/#:~:text=On%20October%2021%2C%202016%2C%20Dyn,of%201.5%20terabits%20per%20second.

[7] Zeek Logs Documentation: http://gauss.ececs.uc.edu/Courses/c6055/pdf/bro_log_vars.pdf

[8] CTU Normal Dataset: https://mcfp.felk.cvut.cz/publicDatasets/CTU-Normal-32/bro/

[9] IEEE IoT Network Intrusion Dataset

https://ieee-dataport.org/open-access/iot-network-intrusion-dataset#files

[10] UCI Machine Learning Repository - Kitsune Network Attack Dataset

https://archive.ics.uci.edu/ml/datasets/Kitsune+Network+Attack+Dataset

[10] Aposemat IoT-23 Dataset - CTU-IoT-Malware-Captures

https://www.stratosphereips.org/datasets-iot23