SOUTHERN METHODIST UNIVERSITY

Information Assurance and Security

Classification of Botnet IOT Network Traffic

Cameron Rosenberger

May 2nd, 2023

Abstract

This report aims to inform on modern distributed software attacks and demonstrate a practical method of defending against IOT botnet attacks. An IOT botnet is a network of devices connected to the internet that have been infected by malware. Typical attacks include Distributed Denial of Service attacks (DDoS), spam attacks, data breaches, etc. A botnet works by spreading malware across as many devices as possible to accomplish some tasks. Tasks depend on the type of attack. Ease of access and effectiveness of these attacks demand a defensive solution to protect IOT devices. While many botnet code bases exist online, this study focuses on Mirai and Gafgyt botnets. This study aims to inform on what malware is, how it attacks, what is at risk, how to detect attacks, and how to defend against attacks. In identifying malware, this study provides a detailed method for detecting botnet infected IOT devices by identifying patterns in internet traffic gathered from a controlled study of 9 infected IOT devices. These patterns are used for supervised training on a machine learning model which can classify a device as benign or infected. Classification of code provides a practical method of monitoring an IOT device for suspicious activity.

Table of Contents

Introduction

This report aims to inform the generally non-technical internet devoice user of the dangers posed by malware, specifically by botnets. With such technical matters as there, it is necessary first to define relevant terms. IOT refers to 'internet of things.' IOT devices are any such piece of technology that is connected to the internet such as phones and computers but also includes devices like a remote baby monitor or a smart doorbell. Any device that can be controlled by a phone is generally an IOT device. These IOT devices are susceptible to malware attacks, with malware being "*any intrusive software developed by cybercriminals (often called hackers) to steal data and damage or destroy computers and computer systems*." [1] The malware being examined in this report is a botnet. *A botnet (short for "robot network") is a network of computers infected by malware that are under the control of a single attacking party, known as the "bot-herder." Each individual machine under the control of the bot-herder is known as a bot.* [2] Botnets examined here are the Mirai and Gafgyt botnets. These two botnets were chosen due to their prevalence and frequency of which they are seen in real internet traffic. "*NetScout research found more than 20,000 unique Mirai samples and variants in the first half of 2019."* [3] Both Mirai and Gafgyt function similarly, although their exploitation methods may differ. Both botnets aim to carry out DDoS attacks and can be used to control any infected device. Once a device is infected by Mirai or Gafgyt, simply powering off the device does not remove the malware, as it is downloaded to non-volatile memory. Further measures must be taken in order to remove malware.

Botnets

As mentioned previously, the two botnets to be examined in this report are the Mirai and Gafgyt botnets. This section aims to detail the origins of these botnets, how they are used, and how they have evolved.

Both botnets discussed here are focused on DDoS attacks. A DDoS attack (distributed denial of service) is designed to flood some receiver with traffic in attempt to overload it with requests. These requests come from each infected device, controlled as a bot, so the network traffic appears to be legitimate. While these requests come from real IOT devices, their traffic has irregularities that can be identified by a machine learning model.

Mirai

Up first for analysis is the Mirai botnet. Mirai gained its popularity from a massive scale attack on Twitter, Spotify, CNN, and many others which took their systems offline temporarily. This attack was recorded as one of the largest DDoS attacks in history. [4] This botnet was first discovered posted to Hack Forums, a popular code discussion forum allegedly by two college students as named by Brian Krebs. [5] These students non-coincidentally started a company dedicated to mitigating the effects of botnets. The Mirai botnet works by scanning a netblock of IP addresses looking for devices like smart home devices. These devices are targeted for the Linux based operating system they are using, more specifically the ARC embedded processor. ARC International is the largest system on chip producer with nearly 1.5 billion chips sold annually. [6] Once a bot has identified an IOT device with an ARC processor, it attempts to brute force attack the password from a table of nearly 60 default passwords used by ARC International. Once the password is successfully uncovered, the bot begins to download its

malware code base to nonvolatile memory. With the code being inside nonvolatile memory, the code remains intact when power is lost. This causes extra difficulty when attempting to remove the malware since simply powering off the device does not remove the malware. Mirai is used to DDoS a specified IP address which could be a WIFI router, a webpage, or a larger server like AWS. [7] While Mirai botnet was released a few years ago, it has yet to stop evolving. In recent news, new evolutions of Mirai named "Okiru" and "Satori" which is Japanese for "get up" and "enlightenment/understanding" respectively. These variants are quite similar to Mirai with the exception in how they gain access to the devices. Variants like these will continue to emerge as code bases for each can readily be found online, allowing any script kiddie to mastermind an attack.

Gafgyt

Gafgyt botnet, sometimes referred to as Bashlite, is very similar to Mirai even sharing a bit of source code. Gafgyt was first discovered in 2014 when a flaw in the bash shell known as ShellShock was exploited. Gafgyt is similar to Mirai in that there is a central server controlling the bots and these bots scan netblocks in search of available targets. Gafgyt however, uses the Common Vulnerabilities and Exposures codes CVE-2017-17215 and CVE-2018-10561 exploits which target Huawei routers to deliver the malware. [8] The Huawei HG532 router has a remote software execution capability. This capability allows one bot to identify a Huawei router and its port 37215 and perform the following commands: [9]

- "wget" – This command fetches the malware payload externally.
- "chmod" – Changes the permission of the downloaded payload.
- "./SCRIPTNAME.sh" – Executes the payload.

Once the device has been infected, various DDoS attack methods can be performed including HTTP flooding, UDP dump, and TCP flood. The source code in Gafgyt for each of these modules have all been seen in Mirai source code. This is part of the reason Gafgyt and Mirai are being examined together in this study.

Identification

Now that the Mirai and Gafgyt botnets have been defined and examined, identification of their presence in IOT devices can be examined. As previously stated, DDoS attacks from Gafgyt and Mirai botnets come from real IOT devices. The network traffic generated by these devices is predictable and can be patterned by a machine learning model. In the section below, I have built a model capable of classifying real internet traffic as normal or malicious. The dataset used comes from a controlled study by the UCI Machine Learning Repository in 2018. [10] The model is a Keras sequential model with four dense layers. The following procedure demonstrates the procedure involved in importing the dataset, cleaning the data, training a model, testing the model, and evaluating the accuracy.

Botnet Network Traffic Classification

by Cameron Rosenberger

Dataset from **Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, D. Breitenbacher,** A. Shabtai, and Y. Elovici 'N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders', IEEE Pervasive Computing, Special Issue - Securing the IoT (July/Sep 2018).

In [14]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import metrics
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.callbacks import EarlyStopping
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
base = '/Users/cameronrosenberger/Downloads/Info Security/Final/archive/'
benign = pd.read_csv(base+'5.benign.csv')
gaf_c = pd.read_csv(base+'5.gafgyt.combo.csv')
gaf_j = pd.read_csv(base+'5.gafgyt.junk.csv')
gaf_s = pd.read_csv(base+'5.gafgyt.scan.csv')
gaf_tcp = pd.read_csv(base+'5.gafgyt.tcp.csv')
gaf_udp = pd.read_csv(base+'5.gafgyt.udp.csv')
mir_a = pd.read_csv(base+'5.mirai.ack.csv')
mir_s = pd.read_csv(base+'5.mirai.scan.csv')
mir_syn = pd.read_csv(base+'5.mirai.syn.csv')
mir_udp = pd.read_csv(base+'5.mirai.udp.csv')
mir_p = pd.read_csv(base+'5.mirai.udpplain.csv')
```

In [3]:
```
benign['type'] = 'benign'
mir_udp['type'] = 'mirai_udp'
gaf_c['type'] = 'gafgyt_combo'
gaf_j['type'] = 'gafgyt_junk'
gaf_s['type'] = 'gafgyt_scan'
gaf_tcp['type'] = 'gafgyt_tcp'
gaf_udp['type'] = 'gafgyt_udp'
mir_a['type'] = 'mirai_ack'
mir_s['type'] = 'mirai_scan'
mir_syn['type'] = 'mirai_syn'
mir_p['type'] = 'mirai_udpplain'
```

In [27]:
```
df = pd.concat([benign, mir_udp,gaf_c,gaf_j,gaf_s,gaf_tcp,gaf_udp,mir_a,mir_
                axis=0, sort=False, ignore_index=True)
print("Number of samples: ",len(df))
```

```
Number of samples:  828260
```

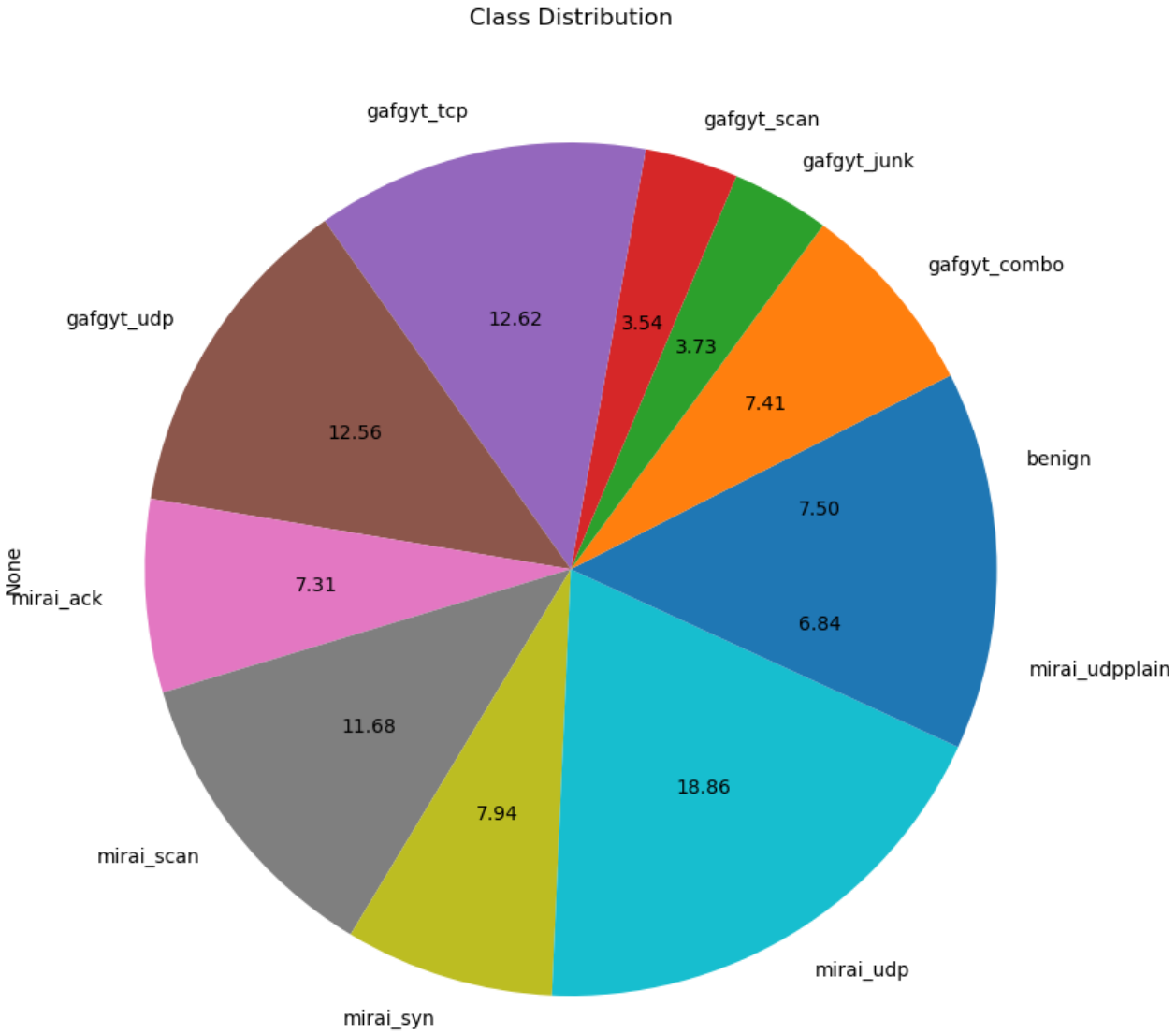
In [18]:
```
df["type"].value_counts()
```

Out[18]:
```
mirai_udp          156248
gafgyt_tcp         104510
gafgyt_udp         104011
mirai_scan          96781
mirai_syn           65746
benign              62154
gafgyt_combo        61380
mirai_ack           60554
mirai_udpplain      56681
gafgyt_junk         30898
gafgyt_scan         29297
Name: type, dtype: int64
```

In [19]:
```
from matplotlib import pyplot as plt

plt.title("Class Distribution")
df.groupby("type").size().plot(kind='pie', autopct='%.2f', figsize=(20,10))
```

Out[19]:
```
<AxesSubplot:title={'center':'Class Distribution'}, ylabel='None'>
```



In [20]:
```
#shuffle the data
sampler=np.random.permutation(len(df))
df=df.take(sampler)
```

In [21]:
```
#dummy encode labels, store separately
labels_full=pd.get_dummies(df['type'], prefix='type')
labels_full.head()
```

Out[21]:

	type_benign	type_gafgyt_combo	type_gafgyt_junk	type_gafgyt_scan	type_gafgy
674211	0	0	0	0	
174091	0	0	0	0	
113938	0	0	0	0	
662995	0	0	0	0	
476999	0	0	0	0	

In [22]:
```
#drop labels from training dataset
df=df.drop(columns='type')
```

In [23]:
```
#standardize numerical columns
def standardize(df,col):
    df[col]= (df[col]-df[col].mean())/df[col].std()

data_st=df.copy()
for i in (data_st.iloc[:,:-1].columns):
    standardize (data_st,i)

data_st.head()
```

Out[23]:

	MI_dir_L5_weight	MI_dir_L5_mean	MI_dir_L5_variance	MI_dir_L3_weight	MI_dir_L
674211	0.360854	-0.672113	-0.679436	0.086156	-(
174091	0.512901	0.868035	1.762154	0.670245	(
113938	0.274451	0.634032	1.633239	0.403487	(
662995	0.911388	-0.672164	-0.679439	0.462733	-(
476999	-1.075703	-0.672164	-0.679439	-1.073401	-(

5 rows × 115 columns

In [24]:
```
#training data for the neural net
train_data_st=data_st.values
#labels for training
labels=labels_full.values
```

In [26]:
```
# test/train split  25% test
x_train_st, x_test_st, y_train_st, y_test_st = train_test_split(
    train_data_st, labels, test_size=0.25, random_state=42)

#  create and fit model
model = Sequential()
model.add(Dense(10, input_dim=train_data_st.shape[1], activation='relu'))
model.add(Dense(10, input_dim=train_data_st.shape[1], activation='relu'))
model.add(Dense(1, kernel_initializer='normal'))
model.add(Dense(labels.shape[1],activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')
monitor = EarlyStopping(monitor='val_loss', min_delta=1e-3,
                        patience=5, verbose=1, mode='auto')
model.fit(x_train_st,y_train_st,validation_data=(x_test_st,y_test_st),
          callbacks=[monitor],verbose=2,epochs=50)
```

```
Epoch 1/50
19413/19413 - 13s - loss: 0.7115 - val_loss: 0.4608 - 13s/epoch - 653us/step
Epoch 2/50
19413/19413 - 12s - loss: 0.3934 - val_loss: 0.3624 - 12s/epoch - 626us/step
Epoch 3/50
19413/19413 - 12s - loss: 0.3242 - val_loss: 0.3206 - 12s/epoch - 611us/step
Epoch 4/50
19413/19413 - 12s - loss: 0.2971 - val_loss: 0.3094 - 12s/epoch - 611us/step
Epoch 5/50
19413/19413 - 12s - loss: 0.2820 - val_loss: 0.2676 - 12s/epoch - 619us/step
Epoch 6/50
19413/19413 - 12s - loss: 0.2718 - val_loss: 0.2760 - 12s/epoch - 603us/step
Epoch 7/50
19413/19413 - 12s - loss: 0.2635 - val_loss: 0.2618 - 12s/epoch - 610us/step
Epoch 8/50
19413/19413 - 12s - loss: 0.2575 - val_loss: 0.2464 - 12s/epoch - 605us/step
Epoch 9/50
19413/19413 - 12s - loss: 0.2526 - val_loss: 0.2425 - 12s/epoch - 608us/step
Epoch 10/50
19413/19413 - 12s - loss: 0.2469 - val_loss: 0.2515 - 12s/epoch - 604us/step
Epoch 11/50
19413/19413 - 12s - loss: 0.2430 - val_loss: 0.2364 - 12s/epoch - 615us/step
Epoch 12/50
19413/19413 - 12s - loss: 0.2408 - val_loss: 0.2471 - 12s/epoch - 615us/step
Epoch 13/50
19413/19413 - 12s - loss: 0.2380 - val_loss: 0.2348 - 12s/epoch - 608us/step
Epoch 14/50
19413/19413 - 12s - loss: 0.2352 - val_loss: 0.2273 - 12s/epoch - 608us/step
Epoch 15/50
19413/19413 - 12s - loss: 0.2326 - val_loss: 0.2376 - 12s/epoch - 605us/step
Epoch 16/50
19413/19413 - 12s - loss: 0.2307 - val_loss: 0.2255 - 12s/epoch - 607us/step
Epoch 17/50
19413/19413 - 12s - loss: 0.2289 - val_loss: 0.2337 - 12s/epoch - 607us/step
Epoch 18/50
19413/19413 - 12s - loss: 0.2281 - val_loss: 0.2410 - 12s/epoch - 614us/step
Epoch 19/50
```

```
19413/19413 - 12s - loss: 0.2259 - val_loss: 0.2253 - 12s/epoch - 612us/step
Epoch 20/50
19413/19413 - 12s - loss: 0.2247 - val_loss: 0.2334 - 12s/epoch - 612us/step
Epoch 21/50
19413/19413 - 12s - loss: 0.2236 - val_loss: 0.2170 - 12s/epoch - 613us/step
Epoch 22/50
19413/19413 - 12s - loss: 0.2228 - val_loss: 0.2180 - 12s/epoch - 606us/step
Epoch 23/50
19413/19413 - 12s - loss: 0.2213 - val_loss: 0.2709 - 12s/epoch - 604us/step
Epoch 24/50
19413/19413 - 12s - loss: 0.2219 - val_loss: 0.2158 - 12s/epoch - 615us/step
Epoch 25/50
19413/19413 - 12s - loss: 0.2184 - val_loss: 0.2220 - 12s/epoch - 608us/step
Epoch 26/50
19413/19413 - 12s - loss: 0.2174 - val_loss: 0.2110 - 12s/epoch - 605us/step
Epoch 27/50
19413/19413 - 12s - loss: 0.2170 - val_loss: 0.2286 - 12s/epoch - 611us/step
Epoch 28/50
19413/19413 - 12s - loss: 0.2173 - val_loss: 0.2192 - 12s/epoch - 604us/step
Epoch 29/50
19413/19413 - 12s - loss: 0.2151 - val_loss: 0.2160 - 12s/epoch - 603us/step
Epoch 30/50
19413/19413 - 12s - loss: 0.2141 - val_loss: 0.2086 - 12s/epoch - 606us/step
Epoch 31/50
19413/19413 - 12s - loss: 0.2129 - val_loss: 0.2169 - 12s/epoch - 609us/step
Epoch 32/50
19413/19413 - 12s - loss: 0.2121 - val_loss: 0.2172 - 12s/epoch - 605us/step
Epoch 33/50
19413/19413 - 12s - loss: 0.2128 - val_loss: 0.2054 - 12s/epoch - 605us/step
Epoch 34/50
19413/19413 - 12s - loss: 0.2121 - val_loss: 0.2099 - 12s/epoch - 606us/step
Epoch 35/50
19413/19413 - 12s - loss: 0.2108 - val_loss: 0.2086 - 12s/epoch - 608us/step
Epoch 36/50
19413/19413 - 12s - loss: 0.2104 - val_loss: 0.2221 - 12s/epoch - 610us/step
Epoch 37/50
19413/19413 - 12s - loss: 0.2096 - val_loss: 0.2130 - 12s/epoch - 609us/step
Epoch 38/50
19413/19413 - 12s - loss: 0.2097 - val_loss: 0.2085 - 12s/epoch - 609us/step
Epoch 38: early stopping
```

Out[26]:
```
<keras.callbacks.History at 0x7fb9a5467b20>
```

In [28]:
```
# metrics
pred_st = model.predict(x_test_st)
pred_st = np.argmax(pred_st,axis=1)
y_eval_st = np.argmax(y_test_st,axis=1)
score_st = metrics.accuracy_score(y_eval_st, pred_st)
print("accuracy: {}".format(score_st))
```

```
6471/6471 [==============================] - 2s 351us/step
accuracy: 0.8706299954120686
```

As seen from the output of the model, an accuracy of 87% has been achieved when classifying network traffic. One issue facing machine learning with botnet network traffic is the lack of available datasets. Since machine learning and botnets are still relatively new concepts, there are few resources available that have produced a dataset of classified network traffic samples. ML models do better with more training data, so accuracy is expected to increase with additional datasets to analyze. This ml model demonstrates its ability accurately classify botnet activity from both Mirai and Gafgyt. This classification technique is one of many applicable defenses that may be employed.

Defense

Now that one method has been established of identifying botnet activity, other commercially used methods should be considered for defense of an IOT device. One particular method is called 'black-holing' or 'sink-holing' network traffic. This means diverting all network traffic to a 'black hole' where it is discarded. A downfall of this method is that legitimate users are blocked as well as the botnet. This defense strategy is a bit archaic in that it is a working solution in blocking botnet traffic, but it blocks all traffic as well, even from an authorized source. A more appropriate method is in configuring the central router to filter all non-essential network traffic. This may be effective at filtering out simple pings from a botnet, but any sophisticated botnet will defeat this method. As seen in the evolution of Mirai and Gafgyt, botnets are becoming increasingly specialized in attacking systems. A method similar to the ml model demonstrated is in intrusion detection systems. These systems are effective at defending against botnet attacks; however, they need a security expert to oversee its actions and often generate false positives. [10] The best industry used practice is in proper configuration of servers. A server admin can explicitly state what resources an application can use and how it

will respond to network requests. This combined with a DDoS specific intrusion detection system provides the greatest chance for a system to stay operational against a large-scale DDoS attack.

Conclusion

As technology progresses, the rise of malicious internet use will only increase. As seen from Mirai, new generations of botnets evolve constantly with hyper tuning to attack a specific application. Without an intentional effort to combat these attacks, servers are left helpless. With minor adjustments to this program, an IOT device owner is able to monitor the network traffic generated by their device and be alerted when a botnet attack is being conducted on their device. In fighting a botnet, identification is the first step. Not knowing about an attack is just like not defending at all. Proper server configuration, firewalls in place, and an intrusion detection system provide a system with the best chance of maintaining operational status when under an attack. This report is not an all-inclusive, comprehensive defense strategy on botnet attacks, but rather a demonstration of the power of AI combined with cybersecurity.

References

[1] Cisco, "Cisco," Cisco, April 2023. [Online]. Available: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjYlNPb2Lv-AhWBl2oFHXnnAl0QFnoECA8QAw&url=https%3A%2F%2Fwww.cisco.com%2Fsite%2Fus%2Fen%2Fis-malware.html&usg=AOvVaw0FTtAtro62usqsjAAfRs31. [Accessed 21 April 2023].

[2] Botnets, "Palo Alto Networks," Palo Alto Networks, April 2023. [Online]. Available: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjluaHq3Lv-AhVdJEQIHX3GDmoQFnoECBsQAw&url=https%3A%2F%2Fwww.paloaltonetworks.com%2Fcyberpedia%2Fwhat-is-botnet&usg=AOvVaw0NzcpU481TDyv5pI0NTf1H. [Accessed 21 April 2023].

[3] TechTarget, "TechTarget," april 2023. [Online]. Available: https://www.techtarget.com/searchsecurity/feature/The-Mirai-IOT-botnet-holds-strong-in-2020. [Accessed 21 april 2023].

[4] Avast, "Avast," 25 October 2018. [Online]. Available: https://blog.avast.com/hacker-creates-seven-new-variants-of-the-Mirai-botnet?_ga=2.189710059499574 67.1646935306. [Accessed 21 April 2023].

[5] CloudFlare, "CloudFlare," CloudFlare, April 2023. [Online]. Available: https://www.cloudflare.com/learning/ddos/glossary/Mirai-botnet/. [Accessed 21 April 2023].

[6] I. Arghire, "SecurityWeek," 16 January 2018. [Online].

Available: https://www.securityweek.com/Mirai-variant-targets-arc-cpu-based-devices/. [Accessed 21 April 2023].

[7] I. Arghire, "Security Week," 16 January 2018. [Online]. Available: https://www.securityweek.com/Mirai-variant-targets-arc-cpu-based-devices/. [Accessed 21 April 2023].

[8] Siddharth Sharma, "Uptycs," 5 April 2021. [Online]. Available: https://www.uptycs.com/blog/Mirai-code-re-use-in-Gafgyt. [Accessed 21 April 2023].

[9] "National Vulerability Database," April 2023. [Online]. Available: https://nvd.nist.gov/vuln/detail/cve-2017-17215. [Accessed 21 April 2023].

[10] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, D. Breitenbacher, A. Shabtai, and Y. Elovici 'N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders', IEEE Pervasive Computing, Special Issue - Securing the IoT (July/Sep 2018).