Botnet Traffic Classification

Kate Stadelman

Abstract

In today's digital age, cyber security has become a primary concern for business. While threats come in many forms, a Distributed Denial-of-Service (DDoS) attack may render a business' digital services unusable, costing millions. It is essential that harmful network traffic is identified and blocked as quickly as possible, without disrupting normal network traffic. This is a fundamental capability of many cyber protection solutions, and in this paper, we utilize the machine learning technique of random forest to identify harmful botnet traffic among normal network traffic.

Introduction

Even in the midst of COVID-19, cyber security remains a primary focus for business. "71% of US CEOs said they are 'extremely concerned' about cyber threats – ahead of pandemics and other health crises (46%)" (PwC 2021). These concerns are warranted given the high cost associated with cyber attacks. The average cost of a Distributed Denial-of-Service (DDoS) attack is $123K for small businesses and $2.3M for enterprises (Kaspersky 2018). DDoS attacks are frequently carried out via botnets.

Figure 1: "Botnet" (Tom-b 2010)

A botnet is defined as "a network of private computers infected with malicious software and controlled as a group without the owners' knowledge" ("Oxford Languages" 2021). Typically, a bad actor will infect a device, often a home computer, with malware that leaves it operating normally to prevent detection. Indeed, these initial devices are not the primary target, but merely a tool for launching a much larger operation. When enough devices are under the attacker's control, he or she may leverage the botnet to perform a DDoS attack, mine bitcoin, send spam, and/or steal personal data.

When it comes to filtering network traffic, it is imperative that cyber protection tools take a surgical approach. Businesses may have a high volume of normal traffic, and incorrectly blocking traffic may be as damaging to

customer experience and sales as an actual attack. This leads us to our research question:

Can we detect botnet activity among normal network traffic?

Data Source & Definitions

Cyber security data sets are notoriously difficult to obtain because there is a risk of disclosing too much information, leaving the authoring organization's network vulnerable to future compromise. Thankfully, CTU University in Czech Republic created CTU-13, a labeled data set of botnet, normal, and background traffic (Garcia 2014). There are thirteen days of data available, each with a different blend of malware, number of infected hosts, and methods of attack. This research utilizes Day 10, which contains ten infected hosts and a DDoS attack over the Internet Control Message Protocol (ICMP) protocol.

Table 2 – Characteristics of the botnet scenarios. (CF: ClickFraud, PS: Port Scan, FF: FastFlux, US: Compiled and controlled by us.)

Id	IRC	SPAM	CF	PS	DDoS	FF	P2P	US	HTTP	Note
1	√	√	√							
2	√	√	√							
3	√			√				√		
4	√				√			√		UDP and ICMP DDoS.
5		√		√					√	Scan web proxies.
6				√						Proprietary C&C. RDP.
7									√	Chinese hosts.
8				√						Proprietary C&C. Net-BIOS, STUN.
9	√	√	√	√						
10	√				√			√		UDP DDoS.
11	√				√			√		ICMP DDoS.
12							√			Synchronization.
13		√		√					√	Captcha. Web mail.

Figure 2: CTU-13 Data Set – Characteristics of the Botnet Scenarios (Garcia 2014)

Data Set Attributes

The CTU-13 data set contains the following attributes for each flow of network traffic. Please note that the source device initiates the communication, driving the protocol, destination IP address, and destination port. The provided traffic labels allow us to train our random forest model. Flows labeled "from botnet" are considered harmful, but those labeled "to botnet" are not.

- Traffic Label
- Start Time
- Last Time
- Botnet Indicator (derived from Traffic Label: "from botnet" = 1, otherwise 0)
- Source Internet Protocol (IP) Address
- Destination Internet Protocol (IP) Address
- Protocol
- Source Port
- Destination Port
- Duration
- Source Packets
- Destination Packets
- Source Bytes
- Destination Bytes
- Source Network Rate
- Destination Network Rate

Network Protocols & Systems

The following are common network protocols and systems. Their functions are listed, as well as typical ports (if any).

- Domain Name System (DNS), Port 53: The "phonebook of the Internet." Translates domain names to IP addresses (e.g. translates google.com to 8.8.8.8).
- Hypertext Transfer Protocol (HTTP), Port 80: Application-layer protocol for hypermedia documents, such as HyperText Markup Language (HTML) and Cascading Style Sheets (CSS). Foundation of the World Wide Web.
- Hypertext Transfer Protocol Secure (HTTPS) over Transport Layer Security (TLS/SSL), Port 443: A secure version of HTTP that uses TLS/SSL as a sublayer.
- Internet Control Message Protocol (ICMP): Supporting protocol of the Internet protocol suite used by network devices, such as routers, for diagnostics and controls.
- Transmission Control Protocol (TCP): Foundational protocol of the Internet protocol suite that uses a three-way handshake to establish a reliable connection.
- User Datagram Protocol (UDP): Core member of the Internet protocol suite used for low-latency, time-sensitive transmissions, such as DNS lookups, with no guarantees for delivery.

Botnet IP Addresses

The CTU-13 data set identifies the following ten source IP addresses as infected hosts. This is somewhat unrealistic in that a DDoS attack would usually be launched using a larger variety of source hosts in order to prevent detection. In our first implementation of random forest, we include source IP address as a feature, and the model heavily relies on it. In a secondary implementation, we exclude source IP address and port to simulate a more true to life environment.

All botnet devices run Windows XP operating system.

- 147.32.84.165
- 147.32.84.191
- 147.32.84.192
- 147.32.84.193
- 147.32.84.204
- 147.32.84.205
- 147.32.84.206
- 147.32.84.207
- 147.32.84.208
- 147.32.84.209

Data Preparation & Feature Analysis

Import Data & Set Data Types

Certain data attributes of the CTU-13 data set appear numeric, but must be treated as strings for our random forest model (e.g. ports). Research on the model indicates that best practice is to explicitly set all data types.

```
# Import Data
data <- read.table("ctu_10_data.csv",sep=",",header = TRUE)

# Explicitly Set Data Types
data <- transform(
  data,
  traffic_label = as.character(traffic_label),
  start_time = as.POSIXct(start_time),
  last_time = as.POSIXct(last_time),
```

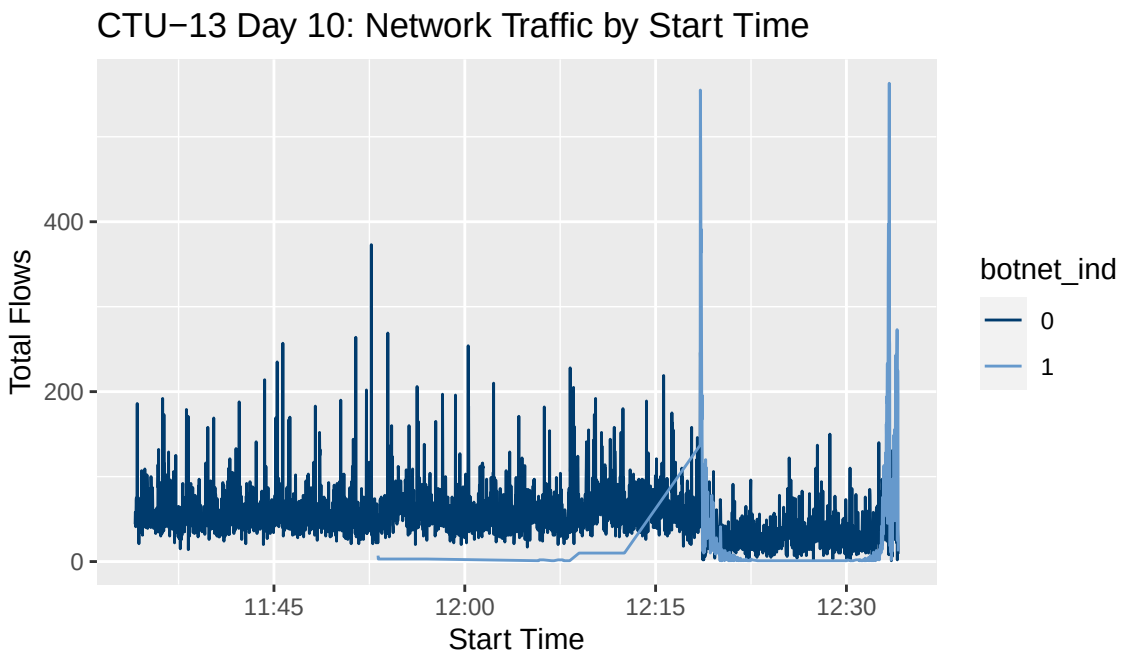
```
  botnet_ind = as.factor(botnet_ind),
  src_address = as.character(src_address),
  dest_address = as.character(dest_address),
  src_port = as.character(src_port),
  dest_port = as.character(dest_port),
  duration = as.numeric(duration),
  src_packets = as.integer(src_packets),
  dest_packets = as.integer(dest_packets),
  src_bytes = as.numeric(src_bytes),
  dest_bytes = as.numeric(dest_bytes),
  src_rate = as.numeric(src_rate),
  dest_rate = as.numeric(dest_rate)
)
```

Network Traffic Over Time

The full CTU-13 Day 10 data set occurs over 4.75 hours and contains 1.3M records. Limitations in memory forced us to reduce the data set to a single hour and 208K records. However, we carefully selected the subset to maintain a consistent overall percentage of botnet records (7.5%).

In the time plot below – with botnet traffic in light blue and normal traffic in dark blue, we can see that the vast majority of traffic is normal (92.5%). Additionally, while there are spikes in botnet traffic at roughly 12:18 pm and 12:33 pm, the majority of time, botnet activity falls within a normal activity range. Unfortunately, identifying all botnet traffic is not as simple as looking for network activity outliers.

```
# Plot Snapshot of Normal & Botnet Traffic
data.time <- data %>% count(start_time, botnet_ind)

(ggplot(data.time) + geom_line(aes(x=start_time, y=n, color=botnet_ind)) +
    scale_color_manual(values= c("0"="#003b6d", "1"="#6699cc")) +
  labs(title="CTU-13 Day 10: Network Traffic by Start Time", x="Start Time",
       y="Total Flows"))
```

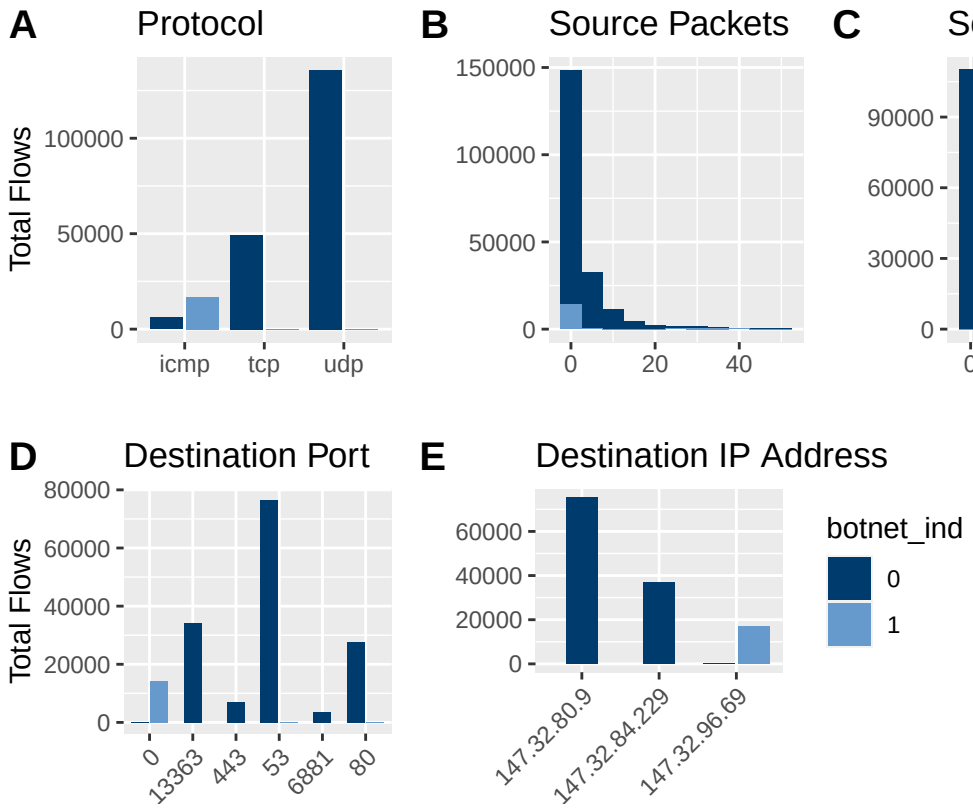


Data Features

In order to better explore features of our data, we construct a grid of visualizations highlighting the similarities and differences between botnet and normal traffic. The CTU-13 data subset contains approximately 20.6K distinct destination IP addresses and 18.6K destination ports, so plots contain only a top handful of each type.

```
# Feature Plot Grid
options(scipen=10L)   # Remove scientific notation from plot
protocols<-c("tcp","udp","icmp")
g1<-ggplot(filter(data, protocol %in% protocols),
           aes(protocol, fill=botnet_ind)) +
  geom_bar(position = position_dodge2(preserve="single")) +
  scale_fill_manual(values=c("0"="#003b6d", "1"="#6699cc")) +
  labs(title="Protocol", x="", y="Total Flows") +
  theme(legend.position="none")

g2<-data %>% filter( src_packets<50 ) %>%
  ggplot(aes(x=src_packets, fill=botnet_ind)) + geom_histogram(binwidth=5) +
  scale_fill_manual(values = c("0" = "#003b6d",
                               "1" = "#6699cc")) +
  labs(title="Source Packets", x="", y="") +
  theme(legend.position="none")

g3<-data %>% filter( src_bytes<2500 ) %>%
  ggplot(aes(x=src_bytes, fill=botnet_ind)) + geom_histogram(binwidth=250) +
  scale_fill_manual(values = c("0" = "#003b6d",
                               "1" = "#6699cc")) +
  labs(title="Source Bytes", x="", y="") +
  theme(legend.position="none")

dest_ports<-c("0","53","13363","80","443","6881")
g4<-ggplot(filter(data, dest_port %in% dest_ports),
             aes(dest_port, fill=botnet_ind)) +
  geom_bar(position = position_dodge2(preserve="single")) +
  scale_fill_manual(values=c("0"="#003b6d", "1"="#6699cc")) +
  labs(title="Destination Port", x="", y="Total Flows") +
  theme(axis.text.x=element_text(angle=45, hjust=1), legend.position="none")

dest_addresses<-c("147.32.96.69","147.32.80.9","147.32.84.229")
g5<-ggplot(filter(data, dest_address %in% dest_addresses),
             aes(dest_address, fill=botnet_ind)) +
  geom_bar(position = position_dodge2(preserve = "single")) +
  scale_fill_manual(values = c("0" = "#003b6d",
                               "1" = "#6699cc")) +
  labs(title="Destination IP Address", x="", y="") +
  theme(axis.text.x=element_text(angle=45, hjust=1))

ggdraw() +
  draw_plot(g1, x=0, y=0.5, width=0.33, height=0.5) +
  draw_plot(g2, x=0.33, y=0.5, width=0.33, height=0.5) +
  draw_plot(g3, x=0.66, y=0.5, width=0.33, height=0.5) +
  draw_plot(g4, x=0, y=0, width=0.33, height=0.5) +
  draw_plot(g5, x=0.33, y=0, width=0.47, height=0.5) +
  draw_plot_label(label = c("A","B","C","D","E"), size=15,
```

```
x = c(0,0.33,0.66,0,0.33), y = c(1,1,1,0.5,0.5))
```



Once again, we indicate botnet traffic in light blue and normal traffic dark blue. The above visualization confirms the botnet DDoS attack was carried out via the ICMP protocol (as indicated in Figure 2) and mainly focused on a single destination IP address and port. Moreover, botnet traffic appears to have a higher than normal packet size and at times, a greater number of packets in comparison to normal traffic.

Methodology & Findigs

Random Forest

Random Forest is an ensemble machine learning method proposed in 1995 by Tim Kam Ho. Particularly adept at managing large data sets with high dimensionality, it is useful for both classification and regression tasks. Training relies on a technique called "bagging," or bootstrapping aggregation: Random samples with replacement are taken from a training set repeatedly, creating numbers of decision trees based on significant features. After training, predictions are made by taking a majority vote of the trees. This "ensemble" method makes random forest resistant to overfitting. The illustration below provides an excellent visual.

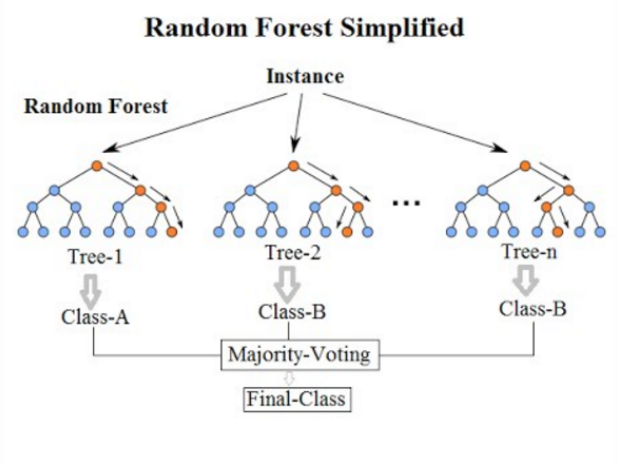


Figure 3: Diagram of a Random Decision Forest (Jagannath 2017)

Network traffic is especially suited to random forest due to the significant size of data sets, high levels of noise, and features that may or may not be relevant depending on the type of cyber attack.

Training & Validation Sets

Before implementing our model, we split our data set into a training set (75%) and test set (25%). Please note, we remove start time and last time attributes because the data set's time of day is arbitrary.

```
# Split Sample into Training & Test Sets
set.seed(123)
sample = sample.split(data$botnet_ind, SplitRatio = .75)
train = subset(data[-c(1:3,15:16)],sample == TRUE)
test = subset(data[-c(1:3,15:16)], sample == FALSE)
```

Model Implementation & Findings

Source IP Address & Port Included

After tuning model parameters mtry, ntree, and maxnodes (which designate the number of variables randomly sampled as candidates at each split, the number of trees, and the maximum number of terminal nodes, respectively), we generate our model using the training set.

```
# Generate Random Forest Model
set.seed(314)
rf.model <- randomForest(
  botnet_ind ~ .,
  data = train,
  mtry = 5,
  ntree = 100,
  maxnodes = 5,
  importance = TRUE
)
rf.model
```

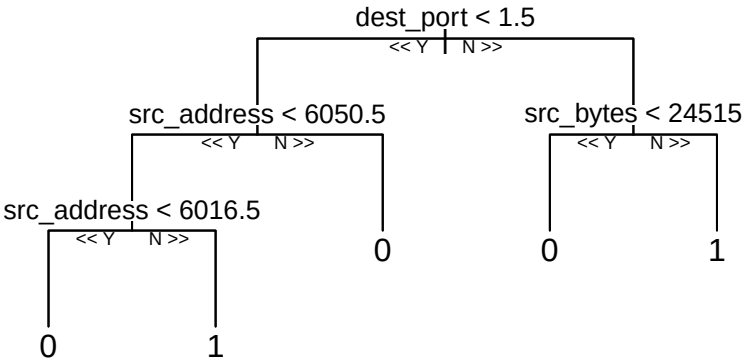
```
##
## Call:
##  randomForest(formula = botnet_ind ~ ., data = train, mtry = 5,      ntree = 100, maxnodes = 5, impo
##                Type of random forest: classification
```

```
##                       Number of trees: 100
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 0.06%
## Confusion matrix:
##        0     1   class.error
## 0 143654     8 0.00005568626
## 1     91 12517 0.00721763959
```

The confusion matrix looks good with minimal error in each class. We further examine our model with additional visualizations and validation.

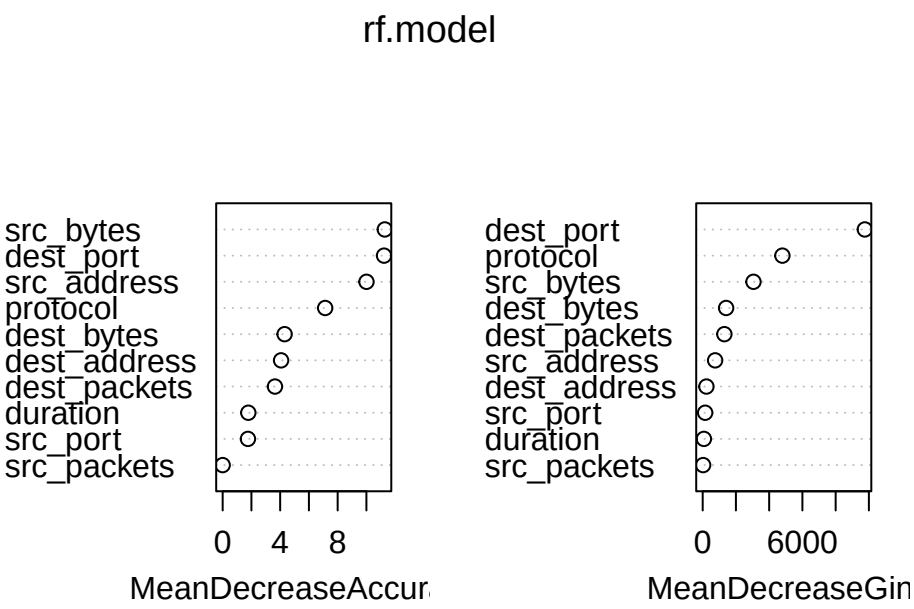
The decision tree overview shows a heavy dependence on source IP address as expected.

```
# Decision Tree Overview
reprtree:::plot.getTree(rf.model)
```



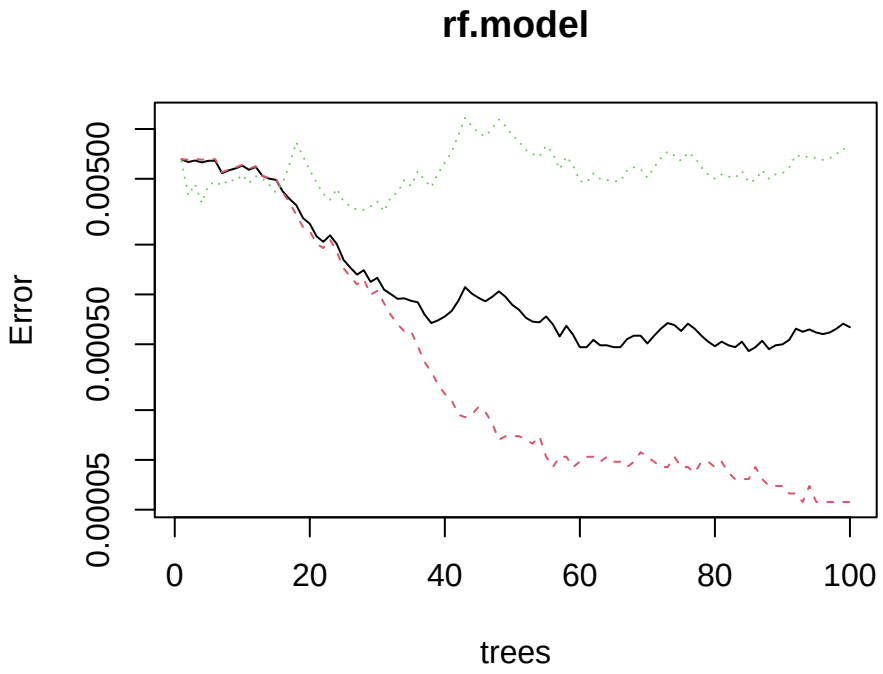
In the next visualization, we review important attributes. The Mean Decrease in Accuracy plot on the left shows the proportion or number of observations that would be incorrectly classified by removing each feature, and the Mean Decrease in Gini plot measures the average gain in purity by splits of a given variable.

```
# Important Attributes
varImpPlot(rf.model)
```



The blow plot of Out-of-Bag (OOB) error displays the OOB error as we add trees. The black line represents the overall OOB error, and the red and green lines represent class (normal vs. botnet) OOB error.

```
# Out-of-Bag (OOB) Error
plot(rf.model, log="y")
```



Finally, we validate our model using our test set. We achieve an incredibly high level of accuracy (99.95%) with a significant p-value.

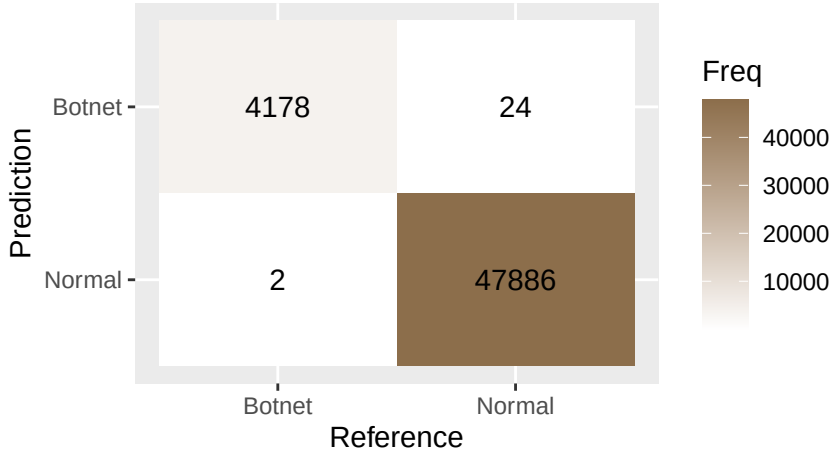
```
# Use Model to Predict Classification of Test Set
pred <- predict(rf.model, newdata=test[-1])
cm <- confusionMatrix(factor(pred), factor(test[,1]), dnn = c("Prediction", "Reference"))
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 47886    24
##          1     2  4178
##
##                Accuracy : 0.9995
##                  95% CI : (0.9993, 0.9997)
##     No Information Rate : 0.9193
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9966
##
##  Mcnemar's Test P-Value : 0.00003814
##
##             Sensitivity : 1.0000
##             Specificity : 0.9943
##          Pos Pred Value : 0.9995
##          Neg Pred Value : 0.9995
```

```
##              Prevalence : 0.9193
##          Detection Rate : 0.9193
##    Detection Prevalence : 0.9198
##       Balanced Accuracy : 0.9971
##
##        'Positive' Class : 0
##
```

The following confusion matrix provides a clear and concise summary of test set outcomes.

```
# Confusion Matrix for Test Set
plt <- as.data.frame(cm$table)
plt$Prediction <- factor(plt$Prediction, levels=rev(levels(plt$Prediction)))

(ggplot(plt, aes(Prediction, Reference, fill= Freq)) +
        geom_tile() + geom_text(aes(label=Freq)) +
        scale_fill_gradient(low="white", high="#8C6E4B") +
        labs(x = "Reference", y = "Prediction") +
        scale_x_discrete(labels=c("Botnet","Normal")) +
        scale_y_discrete(labels=c("Normal","Botnet")))
```



Our results are not quite satisfying given the dependency on source IP address. Would random forest be able to detect botnet traffic in a more realistic attack with a large number of hosts? In the next section, we remove source IP address and port, generate a second model, and find out!

Source IP Address & Port Excluded

We remove source IP address and port from our training and test sets and create a new model. Interestingly, this secondary model achieves almost exact results, but requires an increased number of trees.

```
# Remove Source IP Address & Port from Training & Test Subsets
train.2 = subset(train[-c(2,5)])
test.2 = subset(test[-c(2,5)])

# Generate Random Forest Model
set.seed(314)
rf.model.2 <- randomForest(
  botnet_ind ~ .,
  data = train.2,
  mtry = 5,
  ntree = 200,
```

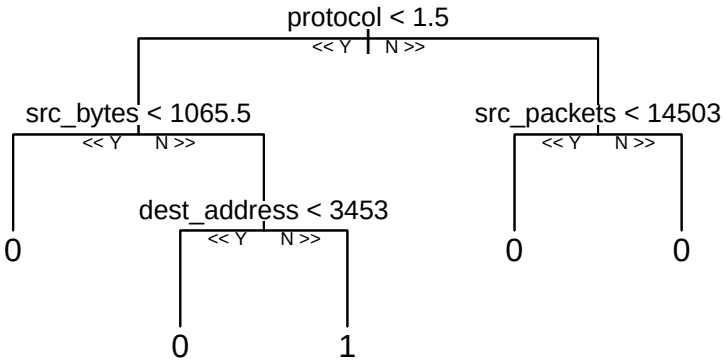
```
  maxnodes = 5,
  importance = TRUE
)
rf.model.2
```

```
##
## Call:
##  randomForest(formula = botnet_ind ~ ., data = train.2, mtry = 5,      ntree = 200, maxnodes = 5, imp
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 0.07%
## Confusion matrix:
##        0     1  class.error
## 0 143633    29 0.0002018627
## 1     80 12528 0.0063451777
```

The second model has fewer false negatives, but more false positives. The class error remains nominal.

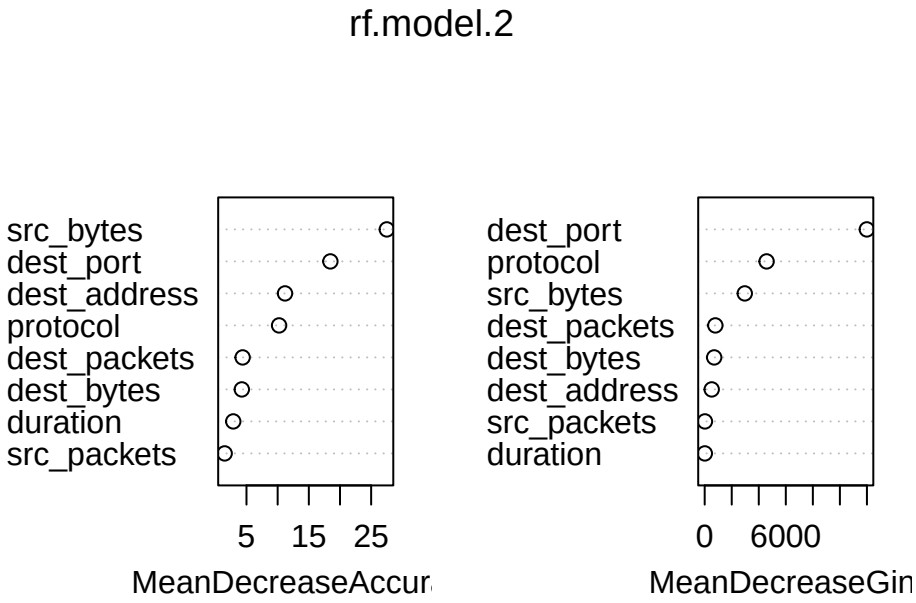
In this model, the decision tree visualization shows protocol, source bytes, and destination IP address as the prominent features.

```
# Decision Tree Overview
reprtree:::plot.getTree(rf.model.2)
```



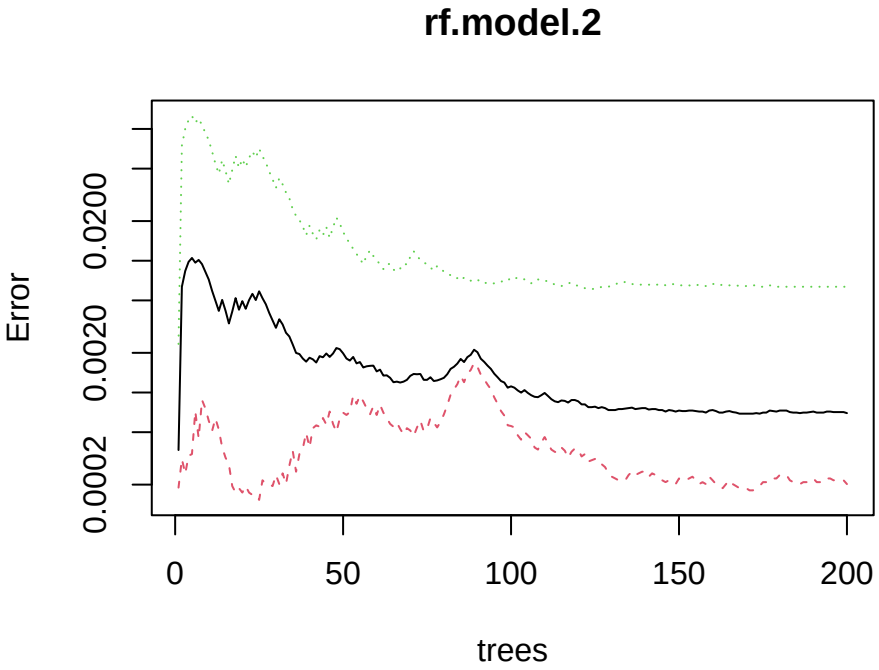
The plot of important attributes reinforces the significance of protocol, source bytes, and destination IP address while also highlighting destination port.

```
# Important Attributes
varImpPlot(rf.model.2)
```



The OOB error plot displays the need for additional trees from the previous model, which minimized OOB error at 100 trees.

```
# Out-of-Bag (OOB) Error
plot(rf.model.2, log="y")
```



In validating our second model, we again achieve an incredibly high level of accuracy (99.94%) with a significant p-value.

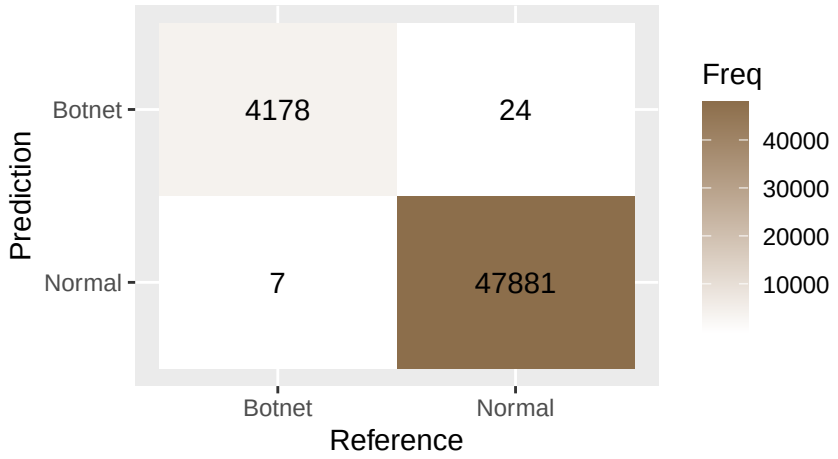
```
# Use Model to Predict Classification of Test Set
pred.2 <- predict(rf.model.2, newdata=test.2[-1])
cm.2 <- confusionMatrix(factor(pred.2), factor(test.2[,1]), dnn = c("Prediction", "Reference"))
cm.2
```

```
## Confusion Matrix and Statistics
```

```
##
##            Reference
## Prediction     0     1
##          0 47881    24
##          1     7  4178
##
##                Accuracy : 0.9994
##                  95% CI : (0.9992, 0.9996)
##     No Information Rate : 0.9193
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.996
##
##  Mcnemar's Test P-Value : 0.004057
##
##             Sensitivity : 0.9999
##             Specificity : 0.9943
##          Pos Pred Value : 0.9995
##          Neg Pred Value : 0.9983
##              Prevalence : 0.9193
##          Detection Rate : 0.9192
##    Detection Prevalence : 0.9197
##       Balanced Accuracy : 0.9971
##
##        'Positive' Class : 0
##
```

While a little less accurate than the initial model, the confusion matrix for our second model shows that random forest is more than capable of identifying botnet traffic, even without source IP address and port!

```
# Confusion Matrix for Test Set
plt.2 <- as.data.frame(cm.2$table)
plt.2$Prediction <- factor(plt.2$Prediction, levels=rev(levels(plt.2$Prediction)))

ggplot(plt.2, aes(Prediction, Reference, fill=Freq)) +
        geom_tile() + geom_text(aes(label=Freq)) +
        scale_fill_gradient(low="white", high="#8C6E4B") +
        labs(x = "Reference", y = "Prediction") +
        scale_x_discrete(labels=c("Botnet","Normal")) +
        scale_y_discrete(labels=c("Normal","Botnet"))
```



Conclusion

As business continues to navigate the perils of the digital age, we must also continue to pursue cyber protection solutions that prevent cyber attacks and safeguard personal data. Using the CTU-13 Day 10 data set, we found that random forest is quite capable of distinguishing between botnet and normal traffic, both with and without source IP address and port.

References

Farnaaz, M. A., N. & Jabbar. 2016. "Random Forest Modeling for Network Intrusion Detection System." *Procedia Computer Science* 89: 213–17. https://www.sciencedirect.com/science/article/pii/S1877050916311127.

Garcia, Grill, S. 2014. "An Empirical Comparison of Botnet Detection Methods." *Computers and Security Journal, Elsevier* 45: 100–123. http://dx.doi.org/10.1016/j.cose.2014.05.011.

Jagannath, V. 2017. "Diagram of a Random Decision Forest." Artwork. CC BY-SA 4.0 via Wikimedia Commons. https://en.wikipedia.org/wiki/Random_forest#/media/File:Random_forest_diagram_complete.png.

Kaspersky. 2018. https://usa.kaspersky.com/about/press-releases/2018_ddos-breach-costs-rise-to-over-2m-for-enterprises-finds-kaspersky-lab-report.

"Oxford Languages." 2021. Dictionary. https://languages.oup.com/google-dictionary-en/.

PwC. 2021. "PwC 24th Annual Global CEO Survey." Survey. https://www.pwc.com/gx/en/ceo-agenda/ceosurvey/2021.html.

Tom-b. 2010. "Botnet." Artwork. CC BY-SA 3.0 via Wikimedia Commons. https://en.wikipedia.org/wiki/File:Botnet.svg.