Analyzing BreachAlarm Password Hacks

Shuddha Chowdhury, Sharmin Jahan, Matthew Weeden

Abstract

Though Internet users are becoming more aware of the security practices of the companies they trust their data with, compromises of this data are becoming more frequent and costly. This paper contributes to the understanding of the user password hacking ecosystem by analyzing the BreachAlarm collection of hacked and published user account data. Namely, we manually categorize the source websites of a large subset of this data and compare breach sizes and the responsible hacker groups across these categories. We found that our source website categories are different in regards to the number of users affected per breach with statistical significance. We find that most hacker groups specialize in one or two types of source websites while one group, "Anon", stands out as generalists. We also find, using the Alexa Top One Million most popular websites list, that popular websites have an estimated 0.2613% probability of experiencing a data breach.

1. Introduction

Breaches to the confidentiality of online user accounts are becoming more potent as Internet users use more Internet services, trusting these companies to protect more of their personal information. The number of companies who are taking on the responsibility to keep user data confidential is increasing as well, and data breaches can translate into significant losses. Verizon, when negotiating the acquisition of Yahoo, cut their buying price by 350 million dollars due to two data breaches that affected more than 1 billion Yahoo users [1]. According to a report done by Forbes Insight and IBM, 46% of US retailers have suffered reputational damage due to a data breach[2].

Hacker groups who successfully access account credentials will sometimes publish lists of the usernames and passwords, possibly for notoriety. In this paper, we describe the state of online user account hacking by analyzing a database of nearly 16,000 publicly-available user account breaches from 2007 to the present maintained by BreachAlarm[3]. The BreachAlarm dataset includes what hacker group claimed the breach, what website or service was breached (also called "sources"), and how many user accounts were compromised. The services reported as having been breached include email services, social media sites, gaming sites, government sites, and foreign web portals among others.

We use this data to answer the following questions: (1) How many users are affected per breach? (2) Which types of websites have the biggest breaches? (3) What hacker groups compromise the most number of user accounts per website type? and (4) What is the annual probability of a breach occurring for popular websites? This final question will be answered using the Alexa Top One Million list of websites (a more detailed description follows in subsection 2.2).

2.1 Related Work

In section 2 part, we describe our methodology for answering these four questions; in particular, we describe how we categorized the breached accounts sources and the statistical methods used for visualization and analysis. In section 3 we give our results, and in section 4 we give a summary and note some immediate and long-term future work that could extend our analysis.

2. Methodology

The work required to answer our four questions comes in two main parts: the categorization of sources and the statistical methods used for visualization and analysis.

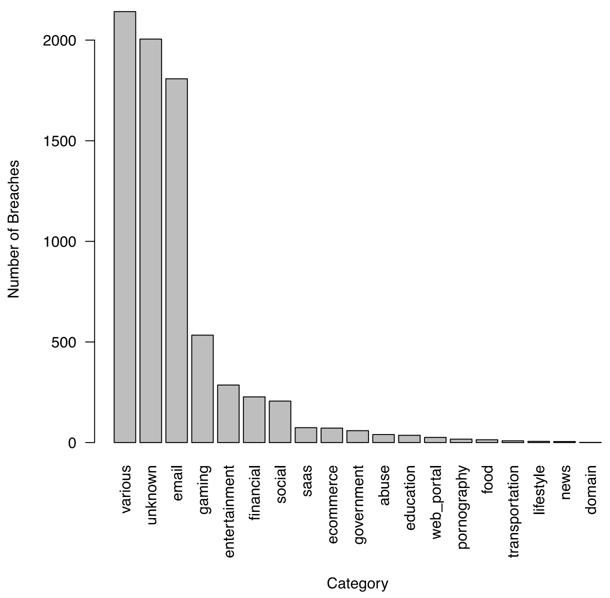


Figure 1: Category Volumes in Breach Dataset

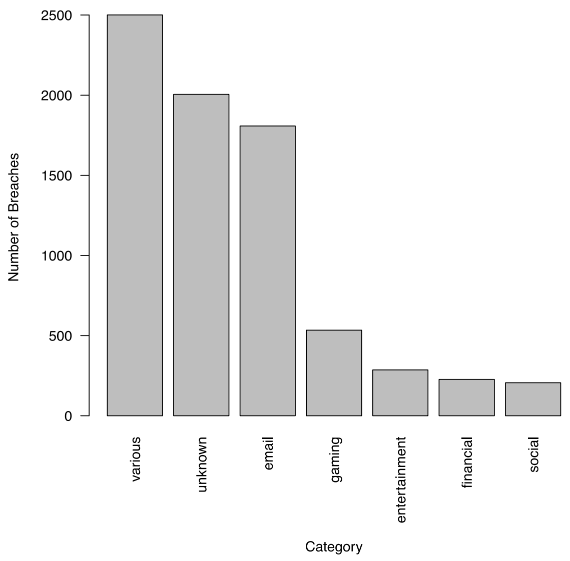


Figure 2: Simplified Category Volumes in Breach Dataset

2.1 Categorization of Sources

We chose to categorize the breach sources assuming that a given source's susceptibility to breach and a hacker group's incentive to compromise these user accounts differ across types of source websites. About 7,500 of the 16,000 breaches in our dataset have a non-domain source value (e.g. "Facebook accounts", "Gmail emails"," Unknown"), and the others contain domains (e.g. "facebook.com", "gmail.com"). Due to time constraints, only the non-domain half was categorized and used in the analysis presented in answering questions (2) and (3).

We based our categorization on prior knowledge and brief web searches. To help carry out this categorization, a sort of "smart categorizer" Python tool was written that would manage the categories and automatically categorize repeated entries.

As we became more familiar with the types of sources in the dataset, five categories stood out as best-defined and most represented: email, gaming, entertainment, financial, and social. Besides these, "unknown" and "various" categories were retained, with each other source category contributing to "various." The amount of representation for each of the original categories can be found in Figure 1, and the same for the simplified categories in Figure 2.

2.2 Visualization and Analysis

To answer question (1), we use the numerical variable "Users." We use a histogram and a boxplot to visualize and describe how many users are affected per breach. For question (2), we create boxplots for each of the simplified source categories, then we establish the statistical significance of the differences between the number of users affected per breach in a series of Wilcoxon tests.

In question (3) we further differentiate within the source categories by determining which hacker groups are responsible for breaches in each category. We use a series of bar plots to determine trends in certain hacker groups' specialization across source categories. To answer question (4), we use the list of 8,000 or so breaches with sources containing domains. Matches in this list are searched for in the Alexa Top One Million list of websites to obtain an underestimate of the proportion of Top Million sites that have been breached since 2007, which is an underestimate of the probability that a given Top Million site has experienced a breach since 2007. These are underestimates assuming that there are many breached user accounts that are not published or that BreachAlarm does not find.

3. Results

The following subsections present the results of our quadripartite analysis with a brief interpretation for each.

3.1 Answering Question (1): How many users are affected per breach?

Figure 3 and Figure 4 show the distribution of breach events over the $\log_{10}$ of the number of users affected per breach, and Table 1 gives the mean, median, and sum of the number of affected users.

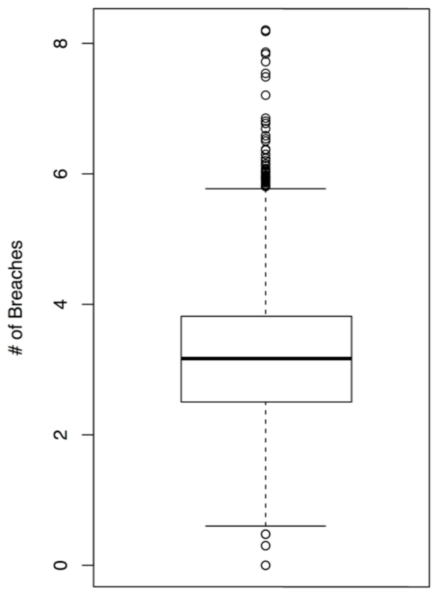


Figure 3: Boxplot of the Log (base 10) Number of Affected Users per Breach

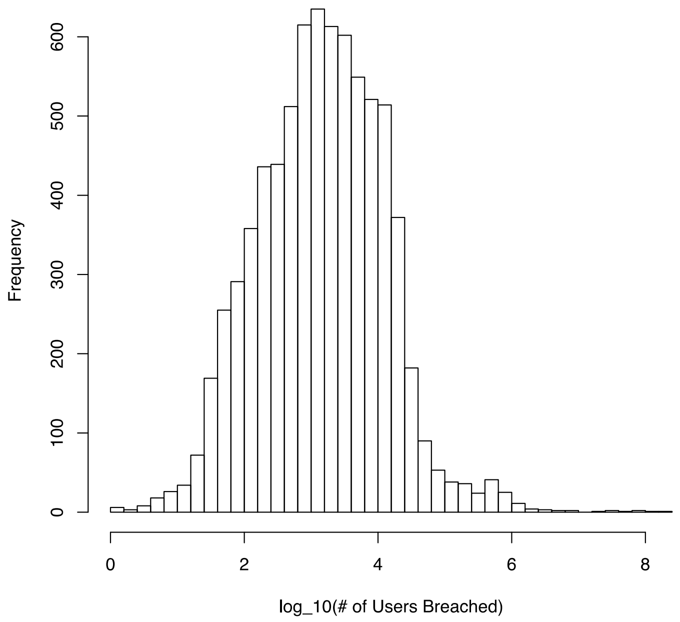


Figure 4: Histogram of the Log (base 10) of the Number of Affected Users per Breach

As can be seen, the mode of the number of affected users is around 1,000 users. The mean number of users is skewed up to 98,335 since there are far more large-breach outliers than small-breach outliers. Only a few breaches compromised upwards of 100 million users and most compromised less than 1,500.

Mean	98,335
Median	1,476
Sum	744,097,496

Table 1: Users Affected Summary Statistics

3.2 Answering Question (2): Which types of websites have the biggest breaches?

The boxplots shown in Figure 5 give an overview of how big a given breach tends to be in each category of source website. As can be expected, many of the categories contain more high-end outliers than low-end. This is probably the case due to the large size of only a handful of sources in a category (e.g. Facebook and Twitter in the social category).

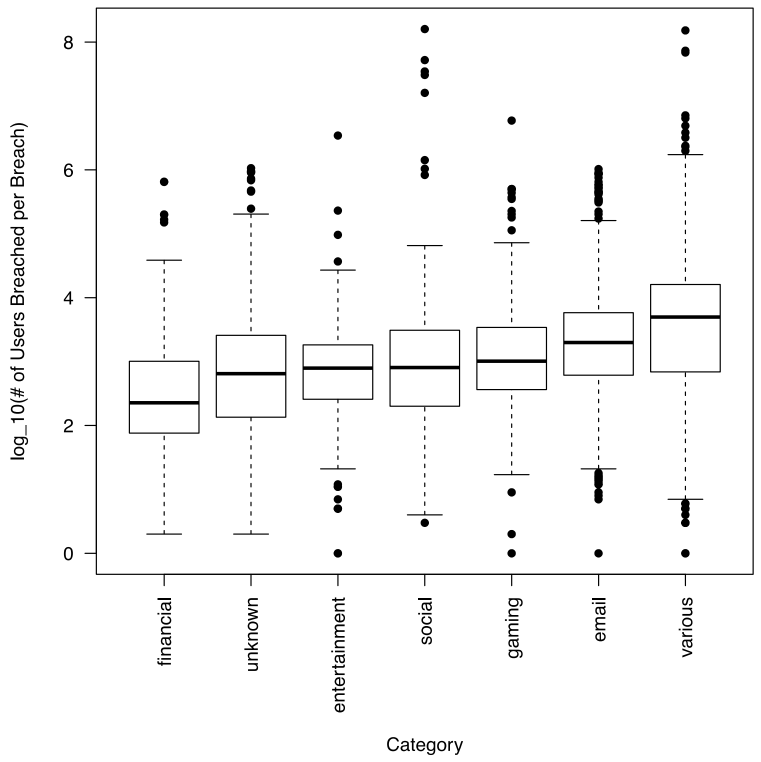


Figure 5: Boxplots of Log (base 10) of the Number of Affected Users per Breach per Category

The difference in the number of affected users across categories could be attributed to the size, vulnerability level, or attractiveness to hacker groups of the source organizations. For instance, while user accounts for financial websites are commonly understood as attractive targets to hacker groups, many of these sites are smaller banks and should have a particular interest in securing their users' data.

Statistical significance of the differences between these categories in the number of affected users per breach was established using 2-way Wilcoxon tests. In each row of Table 2, the log of the number of affected users per breach in the named source category were compared to the same in all other categories. The median of the difference between the two groups is given with a 95% confidence interval. Notice that none of the confidence intervals spans zero, indicating that a clear direction from which each source category differs from the others. Also, the p-value from each test shows that there is a very small chance these differences could be a result of chance rather than true differences between these categories.

Source Category	**Median of the Difference**	**95% Confidence Interval**	**p-value**
various	-0.5919084	[-0.6344785, -0.5489849]	0.00000000000000022
unknown	0.5071905	[0.4611666, 0.5531673]	0.00000000000000022
email	0.1278259	[0.07944465, 0.17632836]	0.0000001634
entertainment	-0.356369	[-0.4573116, -0.2555005]	0.000000000011
financial	-0.763659	[-0.8835589, -0.6416294]	0.00000000000000022
gaming	-0.1338068	[-0.2251794, -0.05518804]	0.0008509
social	0.3218325	[0.1912074, 0.4516591]	0.000001855

Table 2: Two-Way Wilcoxon Results

The financial category has the largest deviation from other groups, indicating a truly different information security situation compared to other websites. It is also significant that both the "unknown" and "various" source categories differ by a median of half an order of magnitude in breach size compared to others. This could be because "unknown" sources tend to be smaller and "various" source organizations belong to underlying groups that might have many users, but where maintaining user data is not as central to their businesses (e.g. automakers, pornography sites, or ecommerce).

3.3 Answering Question (3): What hacker groups compromise the most number of user accounts per website type?

Figure 6 shows the ten hacker groups who compromised the most users overall, and Figure 7 to Figure 13 show the ten hacker groups with the most number of total users affected for each source category.

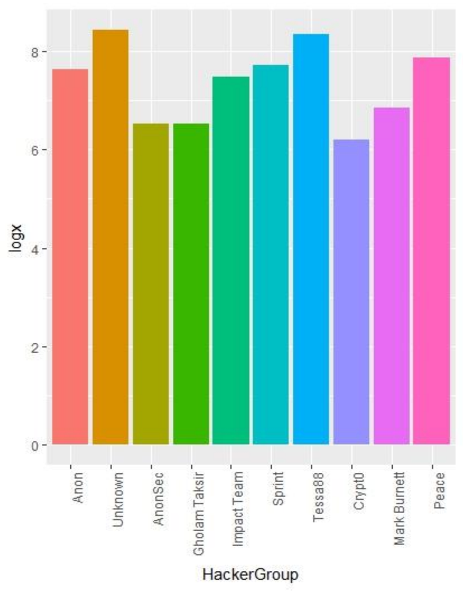
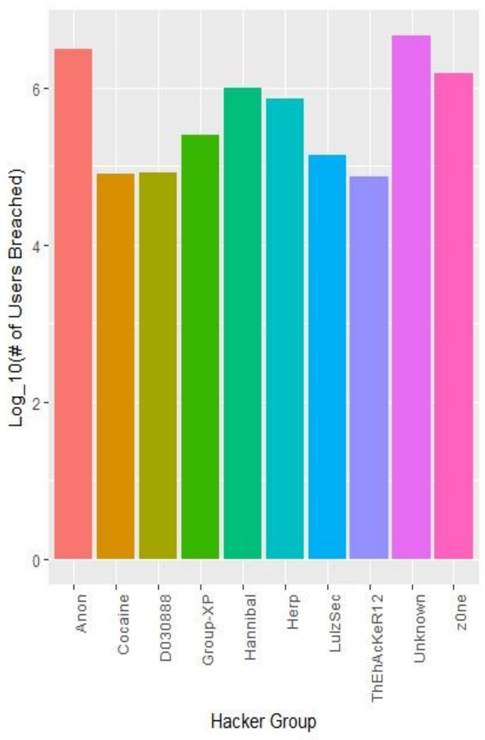


Figure 6: Most Compromising Hacker Groups – Overall



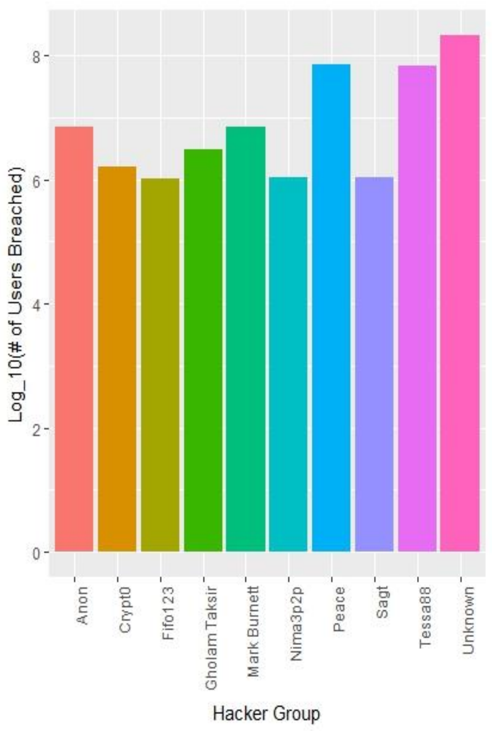


Figure 7: Most Compromising Hacker Groups – Unknown

Figure 8: Most Compromising Hacker Groups - Various

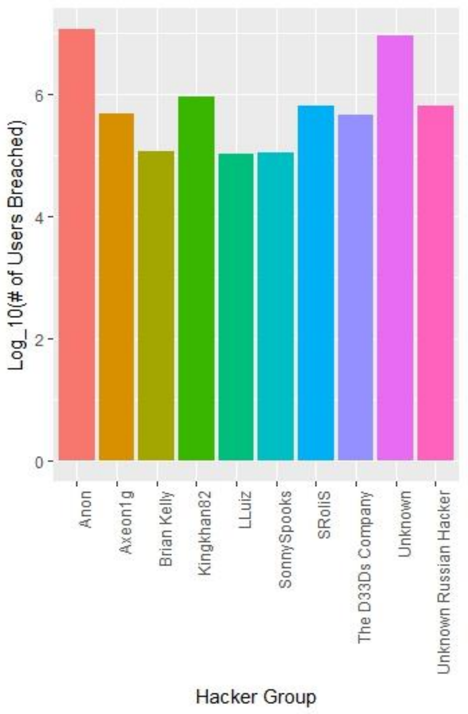


Figure 9: Most Compromising Hacker Groups – Email

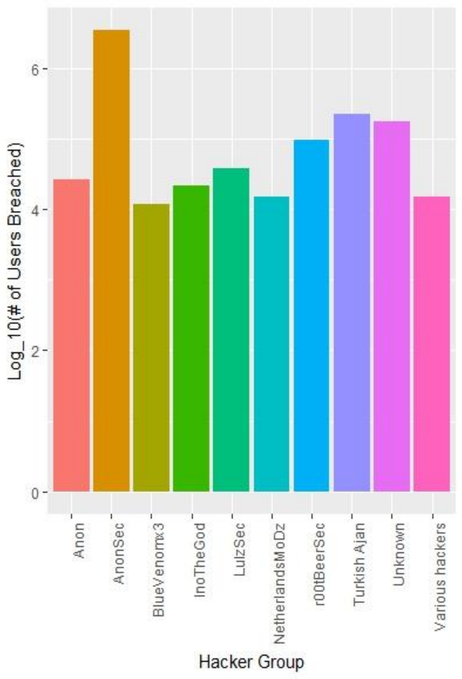


Figure 10: Most Compromising Hacker Groups – Entertainment

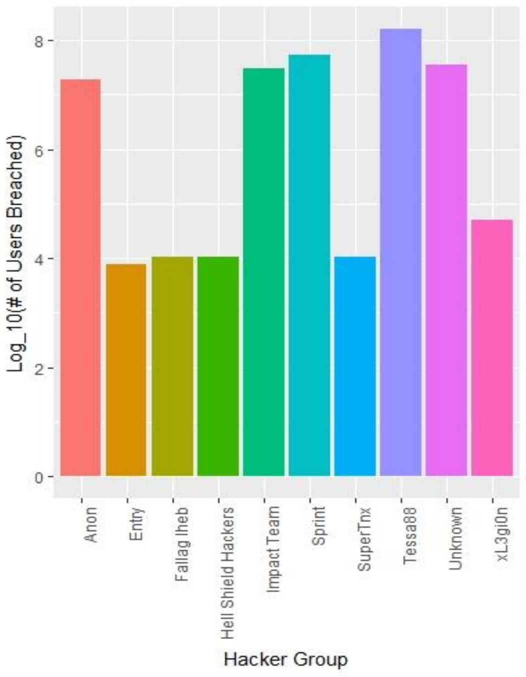


Figure 11: Most Compromising Hacker Groups – Social

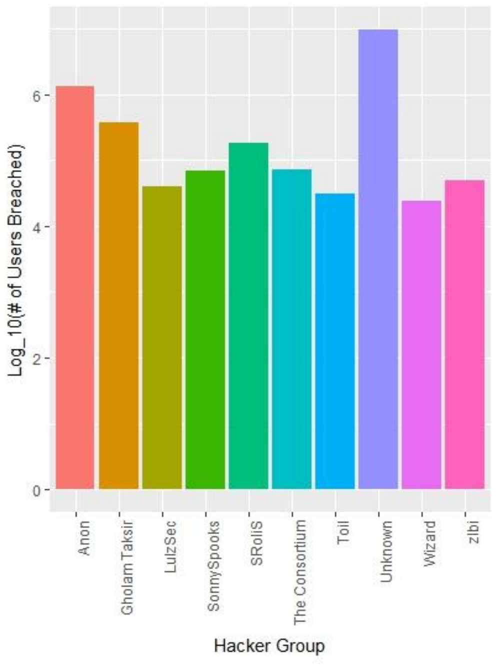


Figure 12: Most Compromising Hacker Groups – Gaming

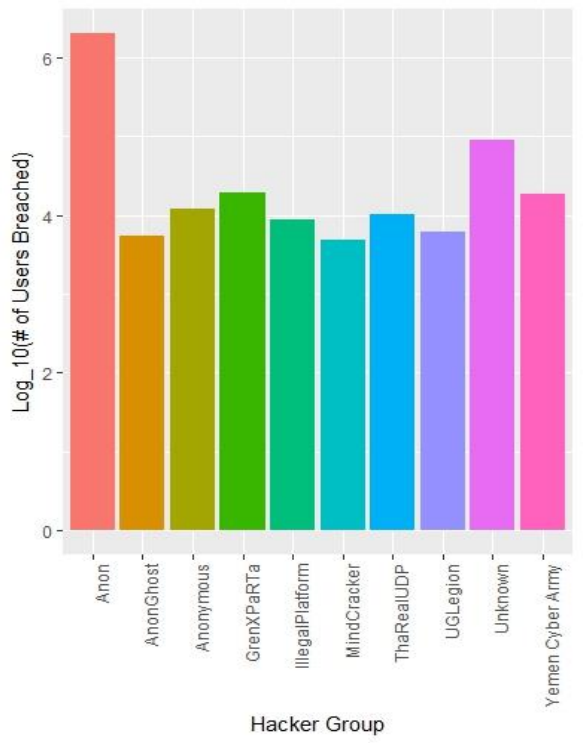


Figure 13: Most Compromising Hacker Groups – Financial

Considering the overall top hacking groups (Figure 6), while most hacker groups can be identified with each breach, a plurality of users have been affected by unknown hacker groups, and there is a significant number of users breached by unknown groups in every category. Besides “Peace” and “Tessa88”, which seems to specialize in “various” and social source categories, “Anon” (short for Anonymous) compromises a significant number of user accounts in each category, standing out as the generalists by a long shot; the next most-occurring group in these top ten lists is “LulzSec”, appearing in three.

3.4 Answering Question (4): What is the annual probability of a breach occurring for popular websites?

Of the breaches with domain sources, 32.52% were found on the Alexa Top Million list (2,613 of 8,036), indicating that only 0.2613% of the Alexa Top Million websites have experienced user breaches detected by BreachAlarm between 2007 and early 2017. While this is clearly an underestimate of the total proportion of breached Alexa Top Million sites, this statistic confirms the common understanding that the probability of a data breach is generally low for any given organization. When making decisions about security investment, this is often outweighed by a usually high potential loss amount if a breach were to occur.

4. Conclusions

Our research is an incremental step in describing the user accounts hacking ecosystem. In conclusion, we will restate our central findings, discuss the limitations of our data and approach, and present some future steps that could be made in this same vein of describing user account hacking.

4.1 Central Findings

In our basic visualization and summary of the breach data number of affected users' variable (question 1), we found that a plurality of breaches had breaches of 1,000 users, most breaches affected less than 1,500 users, and that there are far more large-sized breach outliers than small-sized breaches. Delving into the source website categories (question 2), we found that "various" and email sources experienced the largest breaches while "unknown" and financial sources had the smallest. We also established each source category has a statistically-significant difference in the number of users affected per breach.

Turning to hacker groups responsible for these breaches (question 3), we found that "Anon" stands out as a generalist group, appearing in the top ten most users affected for each of our categories. We also found that a significant number of breaches were accomplished by unknown hacker groups in every source category. In our final analysis (question 4), we found that only 0.2613% of the Alexa Top One Million most popular websites had a user data breach detected by BreachAlarm.

4.2 Limitations

In our decision to not manually categorize the domain-source half of our dataset, have we possibly introduced some bias, analyzing only the sites well known enough to be referred to by a title rather than only a domain name. Also, our categorization method leant itself to considering only the five most well-represented and well-defined types of source websites, though it was obvious that other categories did exist in the dataset (e.g. government sites, education sites, SaaS companies, foreign web portals, etc.).

4.2 Future Work

One clear and accessible continuation of our research would be to include the publication date of these breaches in an analysis. *Have certain types of source websites experience greater increases in user data breaches over time? Are hacker groups active for a long period of time? or do they peak in a consistent way?*

Beyond this, future work could use this data in conjunction with financial loss data (such as the estimates obtained by the Ponemon Institute) to estimate and compare the financial cost each represented organization incurs. *What source website types stand to lose the most from data breaches, and which actually have lost the most?*

References

1. Vindu Goel,Verizon Will Pay $350 Million Less for Yahoo, The New Work Times, Retrieved February 21, 2017, from The New Work Times: https://www.nytimes.com/2017/02/21/technology/verizon-will-pay-350-million-less-for-yahoo.html?_r=0

2. National Cybersecurity Institute, How does a data breach affect your business' reputation? , Cyber Experts Blog,Retrieved February 16, 2016 , from National Cybersecurity Institute: http://www.nationalcybersecurityinstitute.org/general-public-interests/how-does-a-data-breach-affect-your-business-reputation/

3. BreachAlarm, All Data Breach Sources, BreachAlarm, Retrieved April 24, 2011, from BreachAlarm: https://breachalarm.com/all-sources

Appendix

A.1 R code

```
# This R script transforms, visualizes, and test the BreachAlarm dataset
# Shuddha Chowdhury, Matthew Weeden, Sharmin Jahan


# boxplots, histograms, and barplots

db<-read.csv("../data/DataBreach_noDomains_cat.csv",sep=",",quote="'",header=T)
db_s<-read.csv("../data/n_DataBreach_noDomains_cat_simple.csv",sep=",",quote="'",header=T)

# take the log of the Users column
db$UsersLog<-log(db$Users,10)
db_s$UsersLog<-log(db_s$Users,10)

# single boxplot of "# of Users Breached"
pdf("../data/box_total.pdf")
boxplot(db$UsersLog,ylab="# of Breaches")
dev.off()

# stripchart of "# of Users Breached"
pdf("../data/hist_total.pdf")
#stripchart(db$UsersLog,ylab="frequency",pch=10,method="stack",xlab="log_10(# of Users Breached)")
hist(db$UsersLog,breaks=50, xlab="log_10(# of Users Breached)", main="Histogram of log_10(# of Users)")
dev.off()

# sort by category medians
db$S_Cat<-reorder(db$S_Cat,db$Users,median,na.rm=T)
db_s$S_Cat<-reorder(db_s$S_Cat,db_s$Users,median,na.rm=T)

# per category stacked boxplot
pdf("../data/cat_box.pdf")
par(mar=c(9,5,1,1))
```

```
boxplot(UsersLog~S_Cat,method='stack',pch=19,ylab="Number of Users Breached per Breach",xlab="",data=db, las=2)
mtext("Category", side=1, line=7, las=1)
dev.off()

# per category stacked boxplot (simplified categories)
pdf("../data/n_cat_box_simple.pdf")
par(mar=c(9,5,1,1))
boxplot(UsersLog~S_Cat,method='stack',pch=19,ylab="log_10(# of Users Breached per Breach)",xlab="",data=db_s, las=2)
mtext("Category", side=1, line=7, las=1)
dev.off()

# barplot of categories
pdf("../data/cat_barplot.pdf")
par(mar=c(9,5,2,1))
barplot(sort(table(db$S_Cat), decreasing=T), las=2, main='Source Category Frequencies')
mtext('Number of Breaches', side=2, line = 4)
mtext("Category", side=1, line=7, las=1)
dev.off()

# barplot of categories (simple)
pdf("../data/n_cat_barplot_simple.pdf")
par(mar=c(9,5,2,1))
barplot(sort(table(db_s$S_Cat), decreasing=T), las=2, main='Source Category Frequencies')
mtext("Category", side=1, line=7, las=1)
mtext('Number of Breaches', side=2, line = 4)
dev.off()


===============================================================================


# 2-way Wilcoxon tests
db<-read.csv("DataBreach_noDomains_cat_simple.csv",sep=",",quote='"',header=T)
db$logx<-log(db$Users,10)

db$email<-ifelse(db$S_Cat=="email","email","other")
W2.email<-wilcox.test(logx ~ email, data=db,conf.int=T)
W2.email

db$entertainment<-ifelse(db$S_Cat=="entertainment","entertainment","other")
W2.entertainment<-wilcox.test(logx ~ entertainment, data=db,conf.int=T)
W2.entertainment

db$financial<-ifelse(db$S_Cat=="financial","financial","other")
W2.financial<-wilcox.test(logx ~ financial, data=db,conf.int=T)
W2.financial

db$gaming<-ifelse(db$S_Cat=="gaming","gaming","other")
W2.gaming<-wilcox.test(logx ~ gaming, data=db,conf.int=T)
W2.gaming

db$social<-ifelse(db$S_Cat=="social","social","other")
W2.social<-wilcox.test(logx ~ social, data=db,conf.int=T)
W2.social

db$various<-ifelse(db$S_Cat=="various","various","other")
W2.various<-wilcox.test(logx ~ various, data=db,conf.int=T)
W2.various

db$unknown<-ifelse(db$S_Cat=="unknown","unknown","other")
```

```
W2.unknown<-wilcox.test(logx ~ unknown, data=db,conf.int=T)
W2.unknown

=====================================================================================


library(ggplot2)

Top10HackerGroup <- function(CategoryName)
{

db<-read.csv("DataBreach_noDomains_cat_simple.csv",sep=",",quote='"',header=T)

d=subset(db, S_Cat==CategoryName)
d_new<-data.frame(d$Hacker,d$Users)
duserLog<-log(d_new$d.Users,10)
d=aggregate(d_new$d.Users, by=list(HackerGroup=d_new$d.Hacker ), FUN=sum)
d[order(d$x),]
d[order(d$x,decreasing = TRUE),]
fileName<-paste(CategoryName,".csv")
write.csv(d, file = fileName, row.names = TRUE)
s=head(d[order(d$x,decreasing = TRUE),],10)
fileName<-paste(CategoryName,"-2.csv")
s["Category"]<-CategoryName
write.csv(s, file = fileName, row.names = TRUE)
}

Top10HackerGroup("email")
Top10HackerGroup("entertainment")
Top10HackerGroup("financial")
Top10HackerGroup("gaming")
Top10HackerGroup("social")
Top10HackerGroup("unknown")
Top10HackerGroup("various")

db_email<-read.csv("email -2.csv",sep=",",quote='"',header=T)
db_entertainment<-read.csv("entertainment -2.csv",sep=",",quote='"',header=T)
db_financial<-read.csv("financial -2.csv",sep=",",quote='"',header=T)
db_gaming<-read.csv("gaming -2.csv",sep=",",quote='"',header=T)
db_social<-read.csv("social -2.csv",sep=",",quote='"',header=T)
db_unknown<-read.csv("unknown -2.csv",sep=",",quote='"',header=T)
db_various<-read.csv("various -2.csv",sep=",",quote='"',header=T)

mergedata <- rbind(db_email,db_entertainment,db_financial,db_gaming,db_social,db_various,db_unknown)
write.csv(mergedata , file = "mergeData.csv", row.names = TRUE)

mergedata$logx<-log(mergedata$x,10)

dat1=aggregate(mergedata$x,
by=list(HackerGroup=mergedata$HackerGroup,Category=mergedata$Category,value=mergedata$logx ), FUN=sum)
dat1$HackerGroup<-reorder(dat1$HackerGroup,dat1$value,sum)
ggplot(data=dat1, aes(x=HackerGroup, y=value, fill=Category),srt=45) +
   geom_bar(stat="identity")+theme(axis.text.x = element_text(angle = 90, hjust = 1))

--------------------------------------------------------------------------------------------------------------

data1<-data.frame(mergedata$HackerGroup,mergedata$x)
dat1=aggregate(data1$mergedata.x, by=list(HackerGroup=data1$mergedata.HackerGroup), FUN=sum)
dat1[order(dat1$x,decreasing = TRUE),]
s=head(dat1[order(dat1$x,decreasing = TRUE),],10)
```

```
s$logx<-log(s$x,10)
ggplot(data=s, aes(x=HackerGroup, y=logx, fill=HackerGroup),srt=45) +
   geom_bar(stat="identity")+theme(axis.text.x = element_text(angle = 90, hjust = 1))


dat1=aggregate((db_email$x, by=list(HackerGroup=(db_email$HackerGroup), FUN=sum)
db_email[order(db_email$x,decreasing = TRUE),]
s=head(db_email[order(db_email$x,decreasing = TRUE),],10)
s$logx<-log(s$x,10)
ggplot(data=s, aes(x=HackerGroup, y=logx, fill=HackerGroup),srt=45) +
   geom_bar(stat="identity")+theme(axis.text.x = element_text(angle = 90, hjust = 1))+
labs(x="Hacker Group",y="Log_10(# of Users Breached)")


dat1=aggregate((db_financial$x, by=list(HackerGroup=(db_financial$HackerGroup), FUN=sum)
db_financial[order(db_financial$x,decreasing = TRUE),]
s=head(db_financial[order(db_financial$x,decreasing = TRUE),],10)
s$logx<-log(s$x,10)
ggplot(data=s, aes(x=HackerGroup, y=logx, fill=HackerGroup),srt=45) +
   geom_bar(stat="identity")+theme(axis.text.x = element_text(angle = 90, hjust = 1))+
labs(x="Hacker Group",y="Log_10(# of Users Breached)")


dat1=aggregate((db_gaming$x, by=list(HackerGroup=(db_gaming$HackerGroup), FUN=sum)
db_gaming[order(db_gaming$x,decreasing = TRUE),]
s=head(db_gaming[order(db_gaming$x,decreasing = TRUE),],10)
s$logx<-log(s$x,10)
ggplot(data=s, aes(x=HackerGroup, y=logx, fill=HackerGroup),srt=45) +
   geom_bar(stat="identity")+theme(axis.text.x = element_text(angle = 90, hjust = 1))+
labs(x="Hacker Group",y="Log_10(# of Users Breached)")


dat1=aggregate((db_entertainment$x, by=list(HackerGroup=(db_entertainment$HackerGroup), FUN=sum)
db_entertainment[order(db_entertainment$x,decreasing = TRUE),]
s=head(db_entertainment[order(db_entertainment$x,decreasing = TRUE),],10)
s$logx<-log(s$x,10)
ggplot(data=s, aes(x=HackerGroup, y=logx, fill=HackerGroup),srt=45) +
   geom_bar(stat="identity")+theme(axis.text.x = element_text(angle = 90, hjust = 1))+
labs(x="Hacker Group",y="Log_10(# of Users Breached)")


dat1=aggregate((db_social$x, by=list(HackerGroup=(db_social$HackerGroup), FUN=sum)
db_social[order(db_social$x,decreasing = TRUE),]
s=head(db_social[order(db_social$x,decreasing = TRUE),],10)
s$logx<-log(s$x,10)
ggplot(data=s, aes(x=HackerGroup, y=logx, fill=HackerGroup),srt=45) +
   geom_bar(stat="identity")+theme(axis.text.x = element_text(angle = 90, hjust = 1))+
labs(x="Hacker Group",y="Log_10(# of Users Breached)")


dat1=aggregate((db_unknown$x, by=list(HackerGroup=(db_unknown$HackerGroup), FUN=sum)
db_unknown[order(db_unknown$x,decreasing = TRUE),]
s=head(db_unknown[order(db_unknown$x,decreasing = TRUE),],10)
s$logx<-log(s$x,10)
ggplot(data=s, aes(x=HackerGroup, y=logx, fill=HackerGroup),srt=45) +
   geom_bar(stat="identity")+theme(axis.text.x = element_text(angle = 90, hjust = 1))+
labs(x="Hacker Group",y="Log_10(# of Users Breached)")


dat1=aggregate((db_various$x, by=list(HackerGroup=(db_various$HackerGroup), FUN=sum)
db_various[order(db_various$x,decreasing = TRUE),]
s=head(db_various[order(db_various$x,decreasing = TRUE),],10)
s$logx<-log(s$x,10)
ggplot(data=s, aes(x=HackerGroup, y=logx, fill=HackerGroup),srt=45) +
  geom_bar(stat="identity")+theme(axis.text.x = element_text(angle = 90, hjust = 1))+
labs(x="Hacker Group",y="Log_10(# of Users Breached)")
```

A.2 Python code

```
# Password Hacks SEC SP 2017 prject group
# created 4/11/17

import sys
import csv

###########################################################################

f = open("DataBreach.csv", 'rb')
rdr = csv.reader(f, delimiter=',', quotechar='"', quoting=csv.QUOTE_ALL, \
        lineterminator='\n')

nf = open("t.csv", 'wb')
wtr = csv.writer(nf, delimiter=',', quotechar='"', quoting=csv.QUOTE_ALL, \
        lineterminator='\n')

nf2 = open("t2.csv", 'wb')
wtr2 = csv.writer(nf2, delimiter=',', quotechar='"', quoting=csv.QUOTE_ALL, \
        lineterminator='\n')

count = 0
count2 = 0
for r in rdr:
  if "." in r[1] and not " " in r[1]:
    count += 1
    wtr.writerow(r)
    nf.flush()
  else:
    count2 += 1
    wtr2.writerow(r)
    nf2.flush()

print count
print count2

f.close()
nf.close()



###########################################################################
###########################################################################
###########################################################################


# Password Hacks SEC SP 2017 prject group
# created 4/7/17

# This script to automatically categorizes the websites/services in the
#   BreachAlarm dataset
```

```
import sys
import csv

############################################################################

usage = "nope. try this:\nUSAGE: python categorize.py path/to/data.csv " + \
        "path/to/new_data.csv"
if not len(sys.argv) in [4]:
    sys.exit(usage)

f = open(sys.argv[1], 'rb')
rdr = csv.reader(f, delimiter=',', quotechar='"', quoting=csv.QUOTE_ALL, \
                lineterminator='\n')

cats_d = dict()
cat_d = dict()

try:
    f2 = open(sys.argv[2], 'rb')
    rdr2 = csv.reader(f2, delimiter=',', quotechar='"', quoting=csv.QUOTE_ALL, \
                lineterminator='\n')

    # Get categories from file
    for r in rdr2:
        cats_d[r[4][0]] = r[4]
        cat_d[r[1]] = r[4]

    f2.close()
except IOError:
    pass

nf = open(sys.argv[3], 'wb')
wtr = csv.writer(nf, delimiter=',', quotechar='"', quoting=csv.QUOTE_ALL, \
                lineterminator='\n')

print "existing categories:", cats_d

def process_input(cat):
        if cat == "n":
            new_cat = raw_input("Enter new category: ")
            if new_cat[0] in cats_d:
                if new_cat[:2] in cats_d:
                    cats_d[new_cat[:3]] = new_cat
                else:
                    cats_d[new_cat[:2]] = new_cat
            else:
                cats_d[new_cat[0]] = new_cat
            cat_d[row[1]] = new_cat
            print "category", new_cat, "selected.\n================================"
            print "categories:", cats_d
            wtr.writerow(row + [new_cat])
            f.flush()
        elif cat in cats_d:
            cat_d[row[1]] = cats_d[cat]
            print "category", cats_d[cat], "selected.\n================================"
            wtr.writerow(row + [cats_d[cat]])
            f.flush()
        else:
            return process_input(raw_input("Try again:"))
```

```
count = 0

while(True):

    row = rdr.next()

    if count > 0:

        # if this source (website) has been seen before
        if row[1] in cat_d:
            print "category", cat_d[row[1]], "selected.\n================================"
            wtr.writerow(row + [cat_d[row[1]]])
            f.flush()

        else:
            print row[1]
            cat = raw_input("What category should this belong to? Enter n for new category.\n")

            process_input(cat)

    count += 1



##############################################################################
##############################################################################
##############################################################################


# Password Hacks SEC SP 2017 prject group
# created 4/17/17

# This script formats domains and makes a bulk request for info on them from
#   AWIS

import sys
import csv
from awis import AwisApi

##############################################################################

ACCESS_ID = "AKIAI6FLWCIQ2MXU5ALA"
SECRET_ACCESS_KEY = "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"
usage = "nope. try this:\nUSAGE: python awis_domain_data.py path/to/data.csv " + \
        "path/to/new_data.csv"

##############################################################################

if not len(sys.argv) in [3]:
    sys.exit(usage)

f = open(sys.argv[1], 'rb')
rdr = csv.reader(f, delimiter=',', quotechar='"', quoting=csv.QUOTE_ALL, \
                lineterminator='\n')

nf = open(sys.argv[2], 'wb')
wtr = csv.writer(nf, delimiter=',', quotechar='"', quoting=csv.QUOTE_ALL, \
                 lineterminator='\n')
```

```
################################################################################

def filter_dom(d):
    return d

################################################################################

# test python-awis
domains = set(["nginx.com","shadi.com","softlayer.com"])

api = AwisApi(ACCESS_ID, SECRET_ACCESS_KEY)
print 1, api
tree = api.url_info("https://www.paypal.com", "Rank", "LinksInCount")
print 2, tree
print 2, tree.find(".*")
resp = tree.getroot()[0]

print
for r in resp:
    print r.tag, r.attrib

print

#for e in tree.iter(tag="{http://awis.amazonaws.com/doc/2005-07-11}Rank"):
for e in tree.iter():
    print "==", e.tag, e.attrib

count = 0
for r in rdr:
    # skip header row
    if count > 0:
        wtr.writerow(r)
        nf.flush()

    count += 1

f.close()
nf.close()



################################################################################
################################################################################
################################################################################


# Password Hacks SEC SP 2017 prject group
# created 4/17/17

# This script groups categories other than email, gaming, entertainment,
#   social, and financial into the "other" group.

import sys
import csv

################################################################################

usage = "nope. try this:\nUSAGE: python cat_fix.py path/to/data.csv " + \
        "path/to/new_data.csv"
if not len(sys.argv) in [3]:
    sys.exit(usage)

f = open(sys.argv[1], 'rb')
rdr = csv.reader(f, delimiter=',', quotechar='"', quoting=csv.QUOTE_ALL, \
                lineterminator='\n')

nf = open(sys.argv[2], 'wb')
wtr = csv.writer(nf, delimiter=',', quotechar='"', quoting=csv.QUOTE_ALL, \
                 lineterminator='\n')

##############################################################################

cats = set(["email","entertainment","gaming","financial","social","various","unknown"])
count = 0
for r in rdr:
    # skip header row
    if count > 0:
        if r[4] in cats:
            wtr.writerow(r)
            nf.flush()
        else:
            wtr.writerow(r[:4] + ["various"])
            nf.flush()
    else:
        wtr.writerow(r)
        nf.flush()

    count += 1

f.close()
nf.close()
```