# Cyber Case Study Part 2

2023-11-07

## Main Takeaways

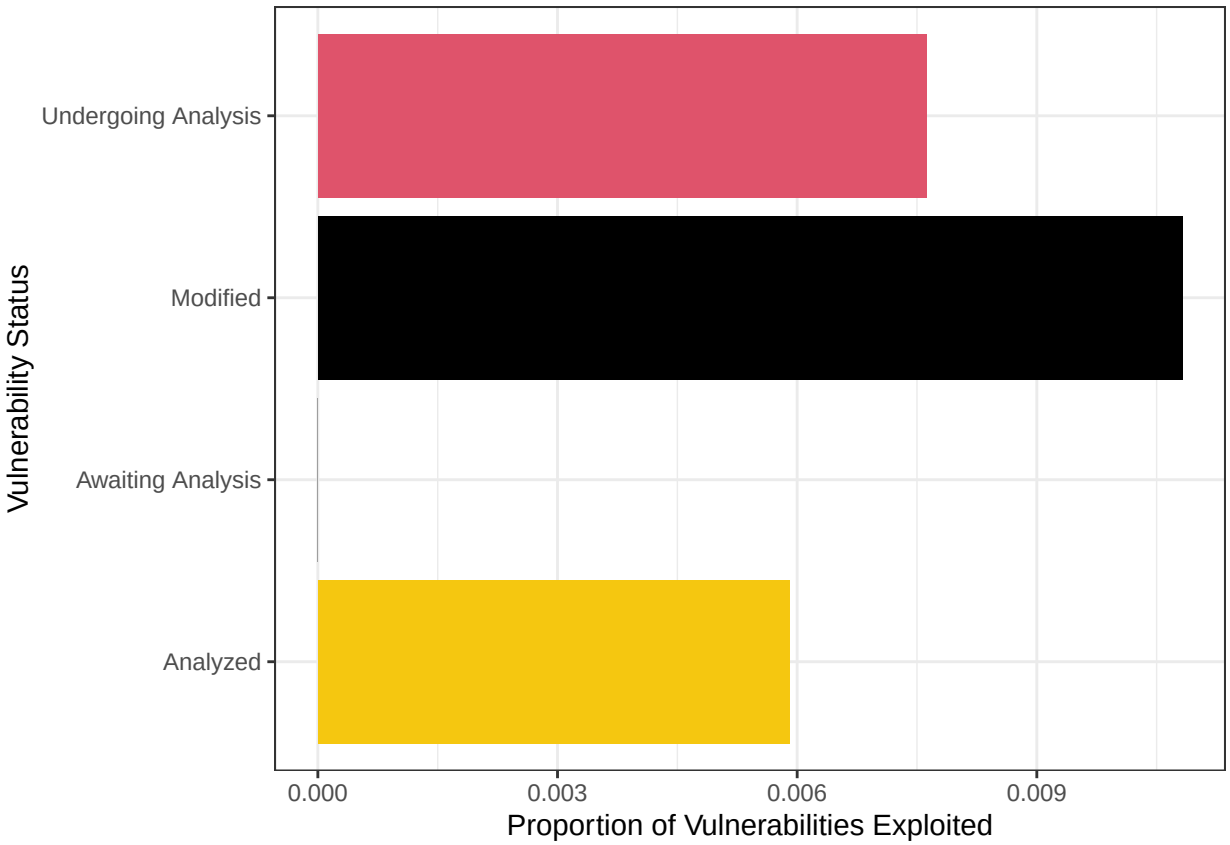
- Vulnerabilities are exploited more often as base score or exploitability score. The relationship between impact score and exploitation is less clear.
- Exploited vulnerabilities are more likely to have a low attack complexity and either low or no privileges required.
- Potential predictor variables (to predict whether or not a vulnerability was required) definitely show some multicollinearity. For example, impact score and exploitability score are both positively correlated with base score.
- Vulnerabilities published or modified more recently have generally been exploited slightly less frequently than older vulnerabilities. Year published and year modified are perfectly correlated (correlation = 1), so we wouldn't want to include them both in a model.

## Supporting Code

```
# load data
cyber_exploits <- readRDS("data/cve.with.exploits.brian.rds")
```

Let's start by looking at what proportion of vulnerabilities have been exploited in each vulnerability status.

```
# calculate proportion of vulnerabilities exploited in each status group
status_plot_data <- cyber_exploits %>%
  group_by(vulnStatus) %>%
  summarize(exploit_rate = mean(exploited), size = n())

# create a bar plot
ggplot(data = status_plot_data, aes(x = exploit_rate, y = vulnStatus)) +
  geom_bar(stat = "identity", fill = 7:10) +
  theme_bw() +
  labs(x = "Proportion of Vulnerabilities Exploited", y = "Vulnerability Status")
```




While it looks like the proportion of vulnerabilities is really different by status, it is worth noting that the x-axis scale is pretty narrow.

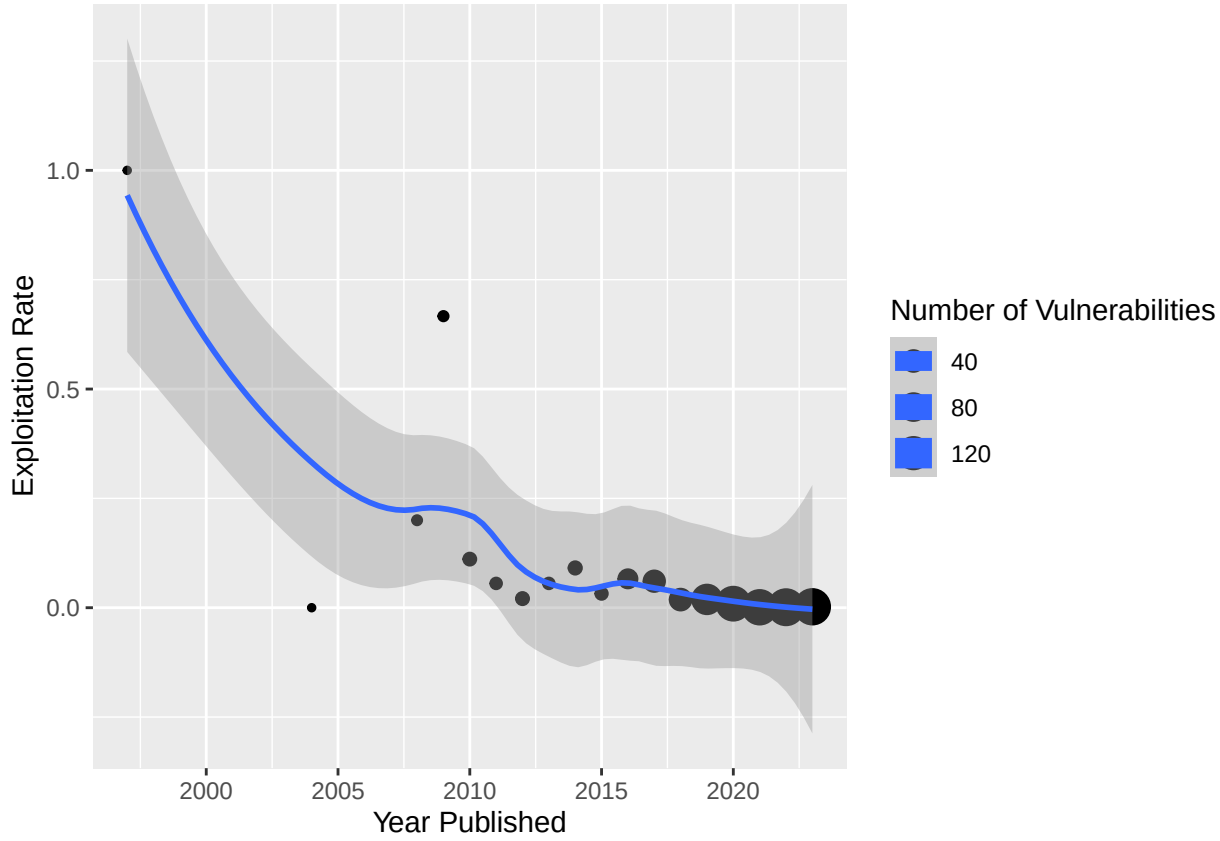
Now, we'll investigate whether the relationship between when a vulnerability was published or modified and whether or not it was exploited.

```
# format published date
cyber_exploits <- cyber_exploits %>%
  mutate(published = as.Date(gsub("T.*", "", published)), lastModified = as.Date(gsub("T.*", "", lastMo
  mutate(year_pub = as.numeric(substr(published, 1, 4)), year_mod = as.numeric(substr(published, 1, 4))

# calculate exploitation rate by year published and year modified
pub_year_plot <- cyber_exploits %>%
  group_by(year_pub) %>%
  summarize(exploit_rate = mean(exploited), size = n())

mod_year_plot <- cyber_exploits %>%
  group_by(year_mod) %>%
  summarize(exploit_rate = mean(exploited), size = n())
```

```
# create plots of date vs exploitation rate
ggplot(pub_year_plot, aes(x = year_pub, y = exploit_rate, size = sqrt(size))) +
  geom_point() +
  geom_smooth() +
  labs(x = "Year Published", y = "Exploitation Rate", size = "Number of Vulnerabilities")
```

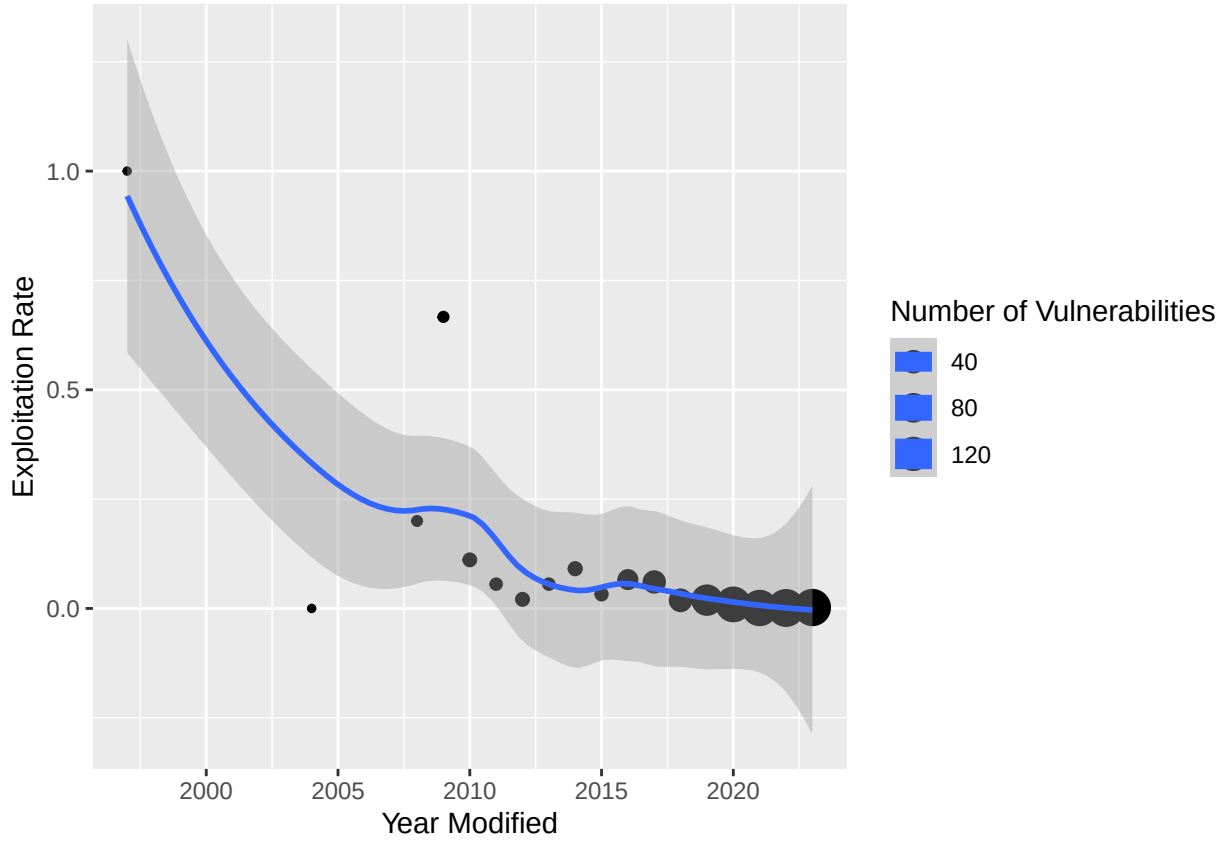
```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
```

```
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.

## ‘geom_smooth()‘ using method = ’loess’ and formula = ’y ~ x’

## Warning: The following aesthetics were dropped during statistical transformation: size
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a ‘group‘ aesthetic or to convert a numerical
##   variable into a factor?
```




```
ggplot(mod_year_plot, aes(x = year_mod, y = exploit_rate, size = sqrt(size))) +
  geom_point() +
  geom_smooth() +
  labs(x = "Year Modified", y = "Exploitation Rate", size = "Number of Vulnerabilities")
```

```
## ‘geom_smooth()‘ using method = ’loess’ and formula = ’y ~ x’

## Warning: The following aesthetics were dropped during statistical transformation: size
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a ‘group‘ aesthetic or to convert a numerical
##   variable into a factor?
```

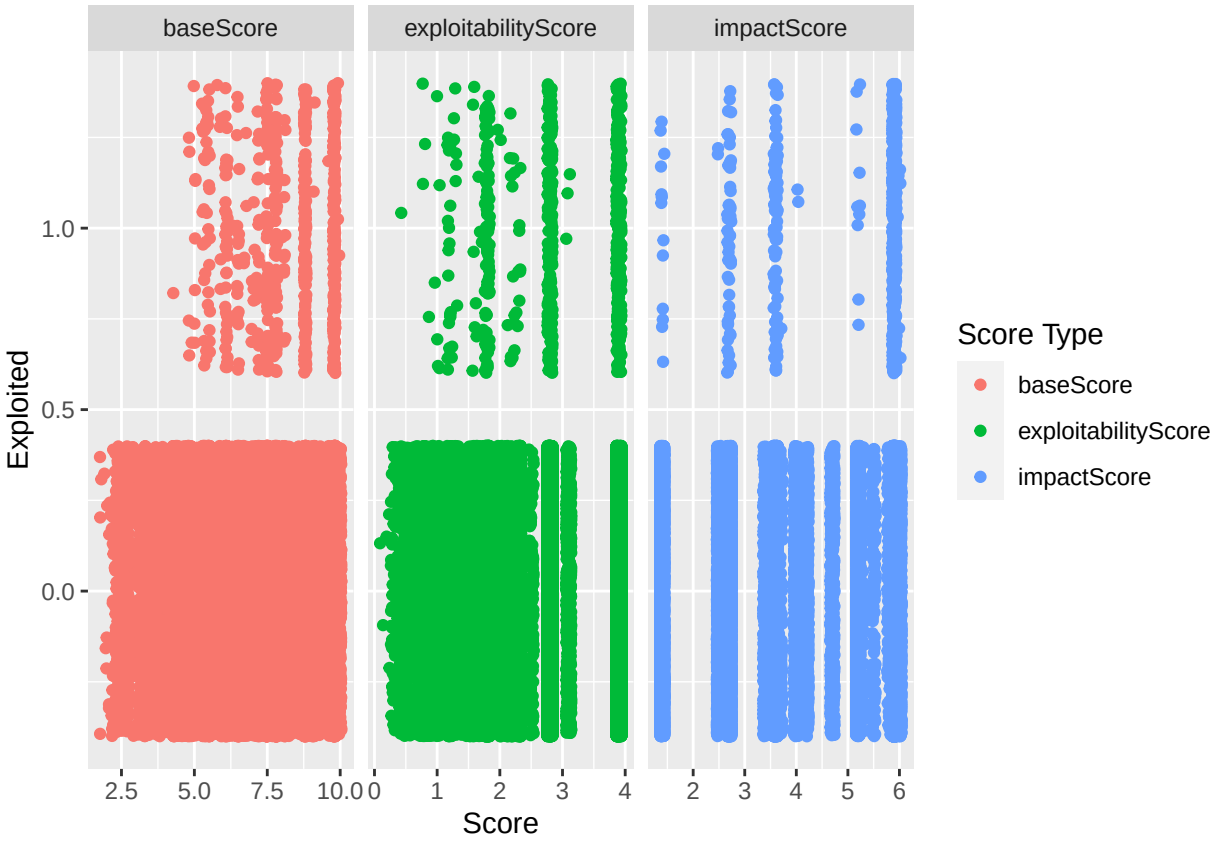



These plots show that the number of vulnerabilities both modified and published over year has generally increased over time. However, the vulnerabilities published or modified more recently have generally been exploited slightly less frequently than older vulnerabilities. Howver, this trend is not super extreme or drastic.

Next, we'll investigate whether/how base score, explotability score, and impact score are related to whether or not a vulnerability was exploited.

```
# put data in a long format so we can facet
long_cyber_df <- cyber_exploits %>%
                 pivot_longer(cols = c(baseScore, exploitabilityScore, impactScore),
                              names_to = 'score_type',
                              values_to = 'score')
```

```
# create plot showing exploitation by each score type
ggplot(long_cyber_df, aes(x = score, y = exploited, col = score_type)) +
  geom_jitter() +
  facet_wrap(~score_type, scales = "free_x") +
  labs(x = "Score", y = "Exploited", col = "Score Type")
```




It looks like vulnerabilities are exploited more often as base score increases. We see a similar trend for exploitability score. THe relationship between impact score and whether or not a vulnerabilit was exploited seems slightly less clear; it looks like there are certain values of impact scores where vulnerabilities are exploited more often (like at a score of around 6 or 3).

Let's create tables to see if vulnerabilities are more or less likely to be exploited based on attack complexity, whether privileges are required, and whether there is user interaction.

```
cyber_exploits %>%
  group_by(attackComplexity) %>%
  summarise(exploitation_rate = mean(exploited))
```

```
## # A tibble: 2 x 2
##   attackComplexity exploitation_rate
##   <chr>                        <dbl>
## 1 HIGH                       0.00576
## 2 LOW                        0.00627
```

```
cyber_exploits %>%
  group_by(privilegesRequired) %>%
  summarise(exploitation_rate = mean(exploited))
```

```
## # A tibble: 3 x 2
##   privilegesRequired exploitation_rate
##   <chr>                           <dbl>
## 1 HIGH                          0.00285
```

```
## 2 LOW                          0.00539
## 3 NONE                         0.00725
```

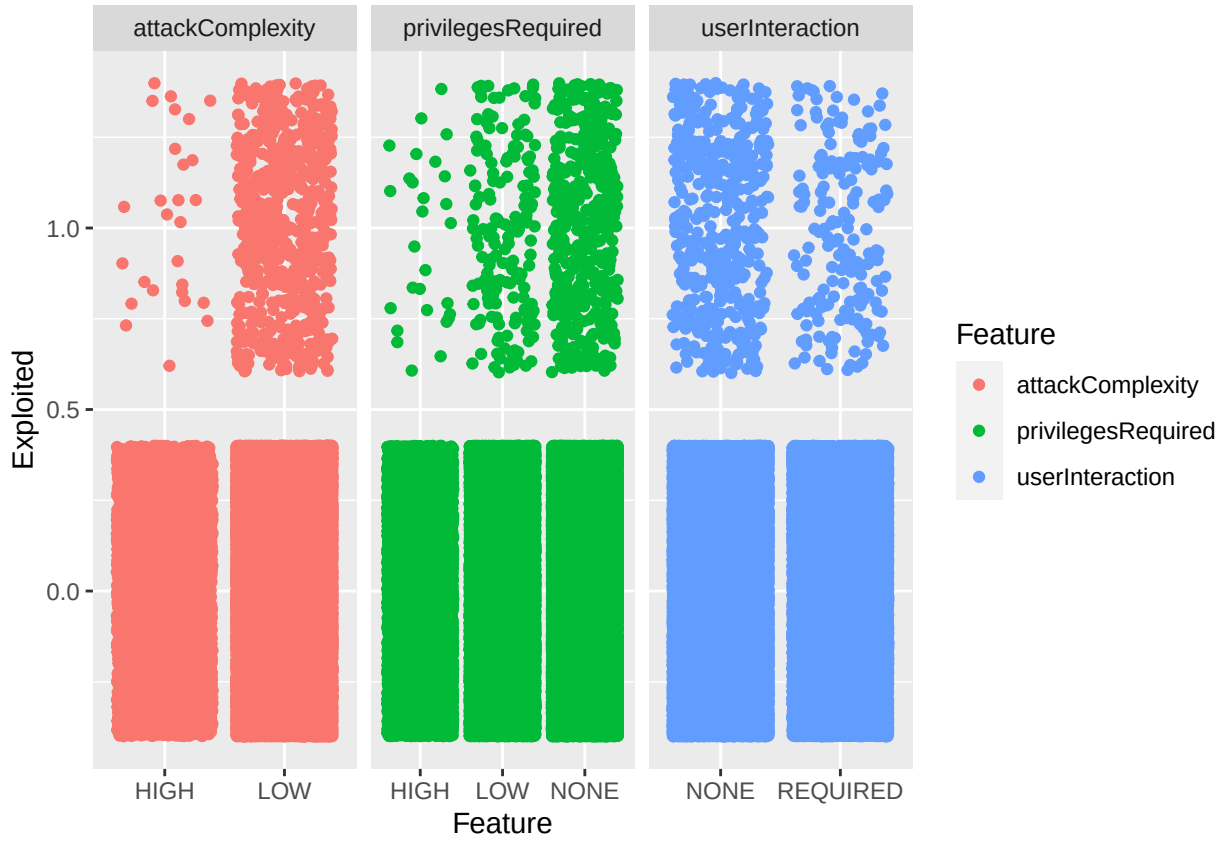
```
cyber_exploits %>%
  group_by(userInteraction) %>%
  summarise(exploitation_rate = mean(exploited))
```

```
## # A tibble: 2 x 2
##   userInteraction exploitation_rate
##   <chr>                       <dbl>
## 1 NONE                      0.00657
## 2 REQUIRED                  0.00560
```

Let's translate these tables into plots to make them easier to see.

```
# put data in a long format so we can facet
long_cyber_df <- cyber_exploits %>%
                 pivot_longer(cols = c(attackComplexity, privilegesRequired, userInteraction),
                              names_to = 'feature_type',
                              values_to = 'feature_value')
```

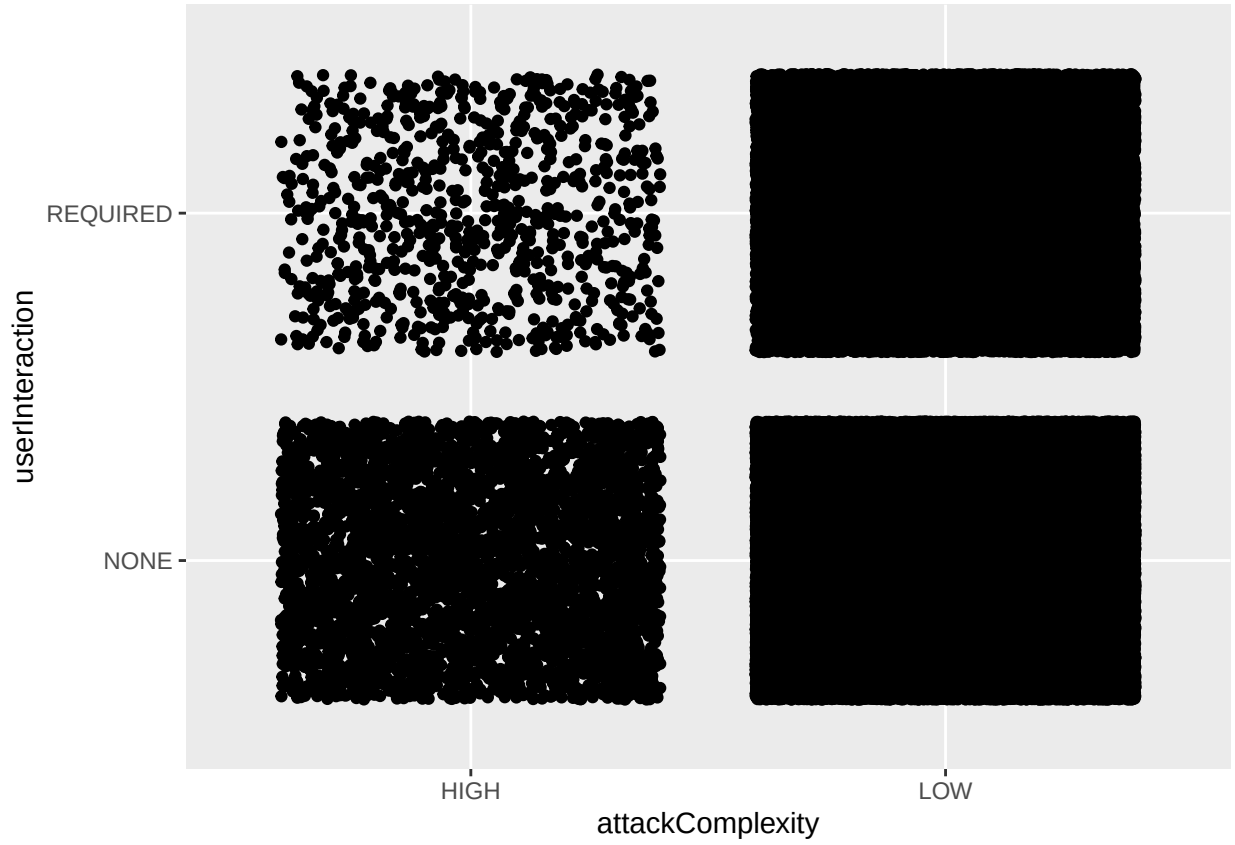
```
# create plot showing exploitation by each score type
ggplot(long_cyber_df, aes(x = feature_value, y = exploited, col = feature_type)) +
  geom_jitter() +
  facet_wrap(~feature_type, scales = "free_x") +
  labs(x = "Feature", y = "Exploited", col = "Feature")
```



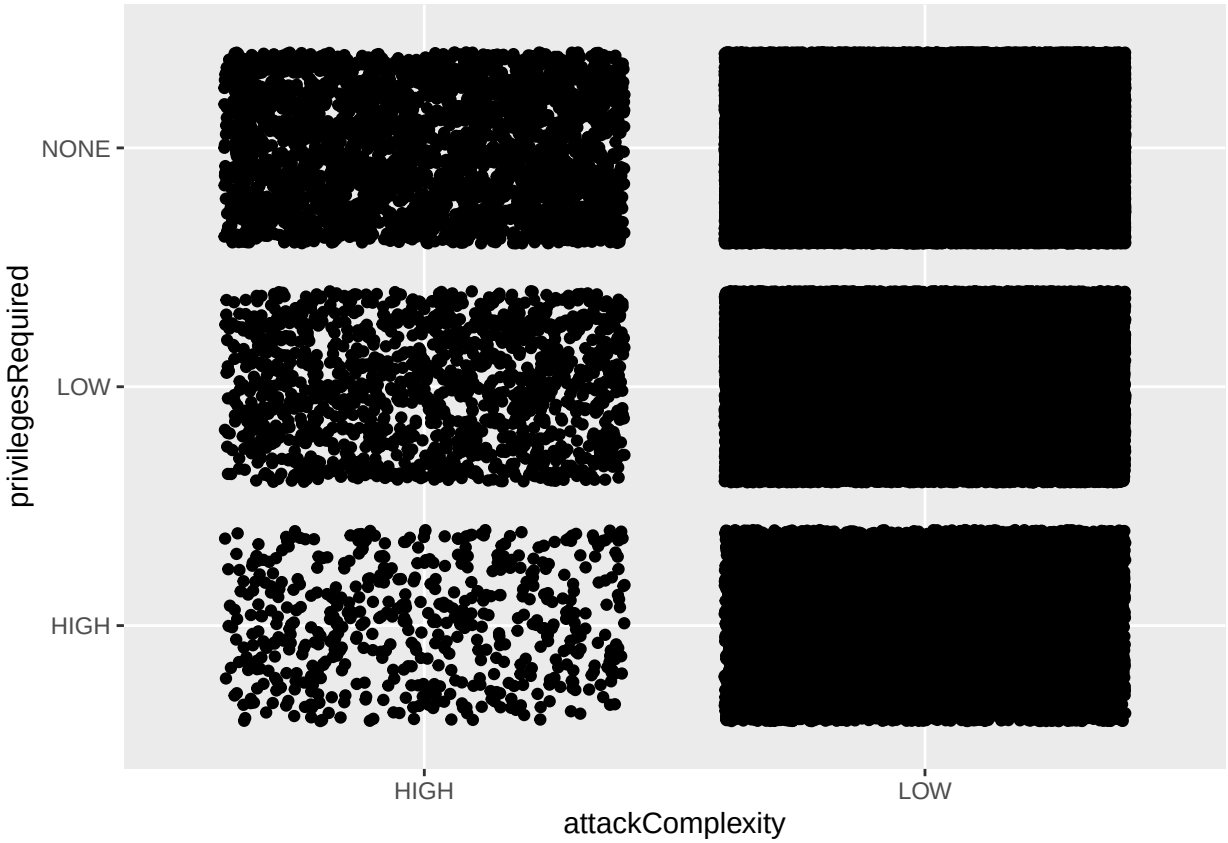
It's clear that exploited vulnerabilities are more likely to have a low attack complexity, either low or no privileges required, and no user interaction. This echos what we saw in the tables above.

Finally, let's investigate how potential predictor variables are related to one another and do some initial modeling. A lot of variables are factors, so I will make calculate some plots in lieu of finding correlations for some variables.

```
ggplot(cyber_exploits, aes(x = attackComplexity, y = userInteraction)) +
  geom_jitter()
```




```
ggplot(cyber_exploits, aes(x = attackComplexity, y = privilegesRequired)) +
  geom_jitter()
```




Definitely looks like there is some sort of relationship between attack complexity, user interaction, and privileges required. It's not exactly what I would expect– for example, I would think an attack would be more likely to be complex if privileges were required.

```
variables_for_correlation <- cyber_exploits %>% select(baseScore, impactScore, exploitabilityScore, yea
cor(variables_for_correlation)
```

```
##                      baseScore impactScore exploitabilityScore    year_pub
## baseScore            1.00000000  0.83265939          0.48711460 -0.02659232
## impactScore          0.83265939  1.00000000         -0.06658155 -0.02403381
## exploitabilityScore  0.48711460 -0.06658155          1.00000000 -0.01315845
## year_pub            -0.02659232 -0.02403381         -0.01315845  1.00000000
## year_mod            -0.02659232 -0.02403381         -0.01315845  1.00000000
##                        year_mod
## baseScore           -0.02659232
## impactScore         -0.02403381
## exploitabilityScore -0.01315845
## year_pub             1.00000000
## year_mod             1.00000000
```

Both impact score and exploitability score have a notable positive correlation with base score. I wonder if the different score types are linear combinations of each other (or otherwise are calculated based on other score types).

Year published and year modified are perfectly correlated.

Now, we'll do some initial modeling. Eventually, we will likely want to run a logistic regression model, since our response is a binary variable.

```r
m1 <- glm(exploited ~ published + lastModified, data = cyber_exploits)
summary(m1)
```

```
##
## Call:
## glm(formula = exploited ~ published + lastModified, data = cyber_exploits)
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.154e-01  1.169e-02   18.44   <2e-16 ***
## published    -2.142e-05  5.763e-07  -37.16   <2e-16 ***
## lastModified  1.021e-05  7.935e-07   12.87   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.006109056)
##
##     Null deviance: 616.13  on 99271  degrees of freedom
## Residual deviance: 606.44  on 99269  degrees of freedom
## AIC: -224360
##
## Number of Fisher Scoring iterations: 2
```

```r
m2 <- glm(exploited ~ published, data = cyber_exploits)
summary(m2)
```

```
##
## Call:
## glm(formula = exploited ~ published, data = cyber_exploits)
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.207e-01  8.355e-03   38.38   <2e-16 ***
## published   -1.666e-05  4.424e-07  -37.65   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.006119192)
##
##     Null deviance: 616.13  on 99271  degrees of freedom
## Residual deviance: 607.45  on 99270  degrees of freedom
## AIC: -224197
##
## Number of Fisher Scoring iterations: 2
```

```r
m3 <- glm(exploited ~ lastModified, data = cyber_exploits)
summary(m3)
```

```
##
## Call:
## glm(formula = exploited ~ lastModified, data = cyber_exploits)
##
```

```
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.726e-01  1.171e-02   14.74   <2e-16 ***
## lastModified -8.707e-06  6.128e-07  -14.21   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.006193989)
##
##     Null deviance: 616.13  on 99271  degrees of freedom
## Residual deviance: 614.88  on 99270  degrees of freedom
## AIC: -222991
##
## Number of Fisher Scoring iterations: 2
```

```
m4 <- glm(exploited ~ published + lastModified + vulnStatus, data = cyber_exploits)
summary(m4)
```

```
##
## Call:
## glm(formula = exploited ~ published + lastModified + vulnStatus,
##     data = cyber_exploits)
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   2.233e-01  1.191e-02  18.747  < 2e-16 ***
## published                    -2.120e-05  5.816e-07 -36.450  < 2e-16 ***
## lastModified                  9.570e-06  8.172e-07  11.711  < 2e-16 ***
## vulnStatusAwaiting Analysis   5.217e-03  2.954e-02   0.177  0.85983
## vulnStatusModified            2.843e-03  1.012e-03   2.809  0.00497 **
## vulnStatusUndergoing Analysis 6.293e-03  3.082e-03   2.042  0.04115 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.006108526)
##
##     Null deviance: 616.13  on 99271  degrees of freedom
## Residual deviance: 606.37  on 99266  degrees of freedom
## AIC: -224366
##
## Number of Fisher Scoring iterations: 2
```

```
m5 <- glm(exploited ~ vulnStatus, data = cyber_exploits)
summary(m5)
```

```
##
## Call:
## glm(formula = exploited ~ vulnStatus, data = cyber_exploits)
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   0.0059002  0.0002599  22.702  < 2e-16 ***
## vulnStatusAwaiting Analysis  -0.0059002  0.0297745  -0.198    0.843
```

```
## vulnStatusModified             0.0049195  0.0009936   4.951 7.39e-07 ***
## vulnStatusUndergoing Analysis  0.0017218  0.0030865   0.558    0.577
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.006205164)
##
##     Null deviance: 616.13  on 99271  degrees of freedom
## Residual deviance: 615.97  on 99268  degrees of freedom
## AIC: -222810
##
## Number of Fisher Scoring iterations: 2
```

## Ideas for Further Analyses

- Analysis of the Vendor organization. Some of the groups have a vendor organization, the most common one is RedHat which is an open source software company that I think does a lot of cybersecurity stuff but not positive.
- We could do some kind of analysis on the email addresses of the source identifier. By the domain name you can see who the source of the vulnerability is, and doing some kind of categorization of these could be interesting but there are so many which might make it hard.