Lab 8: Web - Messaging Security

Jonathan Harijanto

March 7, 2017

CS 373: Defense Against the Dark Arts - Winter 2017

Abstract

The purpose of this blog is to describe my observation from the labs provided by Intel team. This blog will cover the testing methodology and the knowledge that was gain from completing the lab.

I. BLOG

A. What you looked at:

The topic for this week is messaging security. There were one demo and two labs provided by Eric (the instructor) to make us familiar with the concept of security in email. For the demo, we used a new tool called The Regex Coach – an interactive regular expression software, to write an expression that could match a specified string variations. Furthermore, for the first and second lab, we were introduced to another tool called PostgreSQL – an open source object-relational database system, to retrieve an individual data by writing a query on a command line. In the first lab, we had to use PostgreSQL to explore the contents of the data in a given table. On the other hand, for the second lab, we had to write a query in PostgreSQL to classify the data either under SPAM or HAM (non-SPAM) category.

B. How you looked at it & What you found:

1) Lab 1: Data Exploration + DEMO : Regex Coach:

I would like to start the blog by talking about the demo first. I believe the purposes of this demo were to introduce a better approach to writing a regular expression using Regex Coach and to share some tricks in creating an efficient expression. In the demo activity, we had to create a regular expression that could detect the variation of the word "viagra" except the word "Viagra" itself. The variation words given from the instructor were `v | a g r a`, `\/iaGrA`, and `V|4gra`.

In my first trial, I wrote my expression in a hacky way like `v\s\| \w\s\w\s\w\s\w`. Unfortunately, this rule applied to one variation `v | a g r a` only. Thus, I had to write another expression that looks like `(\\\/|v)(i|\|)(a|4)(g)ra` to match with the other two variations. Although my method worked, however, I broke the rule by writing two, instead of one, regular expressions to match all of the variations. Since I did not satisfy with the outcomes, I decided to watch the lecture video and followed Eric's solution. The expression used by Eric was `([vV\\\/]+[ ]*[i|][ ]*[a4][ ]*[gG][ ]*[r][ ]*[aA])`. I need to admit that this expression was indeed "powerful" because it could hit all the variations of the word "viagra". Nevertheless, this expression was too efficient that it also matched the word "Viagra". In the end, I decided to stop working on this RegEx because I already got the concept.

In the first lab, I had to use PostgreSQL that was installed on Xubuntu VM. This tool was completely different with Regex Coach because there wasn't any GUI. The first command that I entered in PostgreSQL was `\d` because I wanted to see the list of tables available. In my case, there was only one table called `message_data`. Thus, my next step was to type `\d message_data` to see all the columns and data types inside the table. The diagram below shows the output printed by PostgreSQL.

Column	Type
mid	character varying(256)
hello_timestamp	int
source_ip	inet
mfrom	character varying(256)
message_size	integer
helo_domain	character varying(256)
msubject	character varying(256)
spam_rules	character varying(1024)
spf_result	character varying(32)
attachment_name	character varying(256)
attachment_hash	character varying(256)
url	character varying(1024)
is_spam	integer

The next step was to answer the list of questions related to `message_data` table. The first question

asked me to find the total numbers (non-distinct) of Records, Source IP, Subjects, Attachment, and URLs. After spending approximately half an hour to refresh my SQL memory, I finally ended up writing a query with this format `SELECT COUNT ... FROM message_data`, where `...` can be substitute with '*' , 'source_ips' , 'msubject' , 'attachment_name' , 'url'. The diagram below shows the detailed answer for question 1.

```
Total records = 100000
The query: SELECT COUNT * FROM message_data;

Total source_ips = 99986
The query: SELECT COUNT (source_ips) FROM message_data;

Total subjects = 100000
The query: SELECT COUNT (msubject) FROM message_data;

Total attachments = 99244
The query: SELECT COUNT (attachment_name) FROM message_data;

Total of urls = 96877
The query: SELECT COUNT (url) FROM message_data;
```

For the second question, I had to find the distinct number of Source IP, Domain, Subjects, Attachments, and URLs. The query was similar with the previous question; the only difference was in the statement `DISTINCT`. Thus, the format of the query to get the answer to question 2 was `SELECT COUNT (DISTINCT ...) FROM message_data;` where `...` can be substitute with 'source_ips' , 'helo_domain' , 'msubject' , 'attachment_name' , 'url'. The diagram below shows the detailed answer for question 2.

```
Distinct source_ips = 25745
The query: SELECT COUNT (DISTINCT source_ips) FROM message_data;

Distinct from domains = 22994
The query: SELECT COUNT (DISTINCT helo_domain) FROM message_data;

Distinct subjects = 16636
The query: SELECT COUNT (DISTINCT msubject) FROM message_data;

Distinct attachment = 76745
The query: SELECT COUNT (DISTINCT attachment_name) FROM message_data;

Distinct urls = 88146
The query: SELECT COUNT (DISTINCT url) FROM message_data;
```

The next question instructed me to find the average message size and the average subject length. Since the data type of `message_size` was already an `int`; thus all I did was to write a query like this `SELECT AVG(message_size) FROM message_data;`. The output showed that the average of message size in `message_data` table was 187560.

Finding the average length of the subject field was tricky because subject's data type was `character varying(256)`, not `int`. Since the `AVG()` couldn't work with `character varying(256)`, I decided to use the help of `CHAR_LENGTH()` function to count the length of the characters in a subject field. In the end, I came

up with a query like this `SELECT AVG(CHAR_LENGTH(msubject)) FROM message_data;`. The output showed that the average of subject field length was 27.

The last question asked for the number of '.zip' , '.rar' , '.pdf' , and '.exe' file extensions from the attachment column in `message_data` table. The only way to get these information was to use a pattern matching technique (`LIKE` operator). The format of the query is as follow `SELECT COUNT (attachment_name) FROM message_data WHERE attachment_name LIKE '%...'`, where `...` can be filled with the word '.zip' , '.rar' , '.pdf' , or '.exe'. The diagram below shows the detailed answer for the final question.

```
Total .zip = 147
The query: SELECT COUNT (attachment_name) FROM message_data WHERE
                attachment_name LIKE '%zip';

Total .rar = 6
The query: SELECT COUNT (attachment_name) FROM message_data WHERE
                attachment_name LIKE '%rar';

Total .pdf = 5179
The query: SELECT COUNT (attachment_name) FROM message_data WHERE
                attachment_name LIKE '%zip';

Total .exe = 1
The query: SELECT COUNT (attachment_name) FROM message_data WHERE
                attachment_name LIKE '%exe';
```

2) Lab 2: Classification Lab:

In the second lab, Eric directed the students to classify the 100,000 real-world messages into either SPAM or HAM (non-SPAM) category. This problem was an open-ended assignment because students were welcomed to write their rule. There are different approaches to group these messages accurately. However, I'm only going to share my personal technique on this blog. The first thing that I did was to initialize every `is_spam` flag to 0. The query that I used to do that was `UPDATE message_data SET is_spam = '0'`.

According to Eric's presentation, analyze the subject field is the easiest way to figure out whether an email is a SPAM or HAM. Thus, I decided to enter a query `SELECT mbsubject FROM message_data` to print all the email's subject in the table. Since there were numerous amount of emails with the subject title 'Our New Stock Pick!', I wrote query `SELECT COUNT (msubject) FROM message_data WHERE msubject LIKE 'Our New Stock Pick!';` to count the total number of these email. I was shocked when the result showed there were 61862 (out of 100000) emails with a subject 'Our New Stock Pick!'.

Next, I was curious about the total number of source IPs that sent those 60000 emails because it's impossible to be done by a single source IP. Therefore, I wrote another query `SELECT COUNT (DISTINCT source_ip) FROM message_data WHERE msubject LIKE 'Our New Stock Pick!';` to count the number of different source IPs that send those SPAM emails. The result revealed that there were 10684 (our of 25745) source IPs that sent this kind of message.

Now that I knew the exact number of the origin IPs that sent the SPAM messages, I decided to classify the list of SPAM/HAM messages based on source IP. The final query that I entered was `UPDATE message_data SET is_spam = '1' WHERE source_ip IN (SELECT DISTINCT source_ip FROM message_data WHERE msubject LIKE 'Our New Stock Pick!');`. Essentially, I used a nested query where it grabbed the list of 10000 IPs first then passed in into the `IN()` clause.

After I had finished typing the final query, there was a little notification that said 65786 data had been updated. In other words, my rule was able to filter out 65786 SPAM emails from 100000 raw emails. Even

though I did not reach 70000-ish email like the students in the video, however, I'm pretty confident that my result will have no false positive.

II. Conclusion

I learned a lot of new kinds of stuff about messaging security this week. First of all, I learned some new tools like ProgreSQL and Regex Coach. Also, I got a chance to refresh my memory about SQL queries command because it's been a while since I deal with a database problem. Lastly, I discovered new tricks in writing a regular expression because I've never exposed with a complicated regex before.