Vulnerability detection for web applications

Mingliang Wang

April 29, 2019

1 Problem

Blind SQL injections are time-based SQL attacks that ask web applications to sleep for a specified amount of time. Typically, the response time of vulnerable web applications will be affected by such attacks while the response time of safe web application will not. Another factor that influences the response time is the network delay which is caused by the communication between a client and a server.

Considering the above facts, we can build models for the response time of safe and vulnerable web applications. For safe web applications,

$$\text{Resonse time} = \text{Network delay}, \tag{1}$$

while for vulnerable web applications, we have

$$\text{Resonse time} = \text{Network delay} + \text{Sleep time} \tag{2}$$

where "Sleep time" is the amount of time that blind SQL injection asks. As indicated by the model (1) and (2), the response time of vulnerable web applications is typically longer than that of safe ones. Thus, time-based algorithms can be used to discriminate if the web application is vulnerable or not. The time-based algorithm can be conducted as follows:

1. send out an attack request (require the application to sleep) and a normal request to the application

2. record the response times of the two requests

3. compare the response times, if the attack response time is significant longer than that of the normal request, we say the web application is vulnerable.

However, one can get false positive results when the network delay is a stochastic phenomenon. For example, in a safe application (1), the response time of a attack request equals the random variable "Network delay"(e.g. 5s), while the response time may equal to 1s. Then we will decide the safe application as a vulnerable one which contradict the truth.

To deal with the false positive problem, the challenge is to design a time-based algorithm such that the false positive rate is less than 10^{-6} and the test should be completed within a few seconds.

The possible solutions are based on hypothesis tests in which the null hypothesis is

$$H_0: \text{ "the url is safe"}$$

and the alternative hypothesis is

$$H_a: \text{"the url is vulnerable"}.$$

The false positive rate is the probability of type I error. To make false positive rate less than 10^{-6}, we set the significance level $\alpha = 10^{-6}$. When the p value $p < 10^{-6}$, we reject H_0, otherwise, we accept it.

Outline The remainder of this report is as follows. Section 2 describe the data of response times for network delays. Section 3 presents two time-based algorithms for vulnerability tests. Section 3.1 gives a t test approach to dealt with false positive cases. Section 3.2 is a χ^2 test approach that only require one request. Finally, Section 4 gives the conclusions.

2 data description

An log or dataset for understanding the network delay is stored in "network_delay.txt" which composed by two columns: column 1 is urls which are in string type and column 2 is the response times which are real numbers. An example of the log is like this:

$$\underbrace{\text{http://127.0.0.1:5000/safe/7/page?id=1}}_{\text{url}} \quad \underbrace{4.49983024597168}_{\text{response time}},$$

where /7/ is a number randomly assigned for item number in reach request, "safe" denotes the safe url which can be changed to "vulnerable" when we request the vulnerable url.

The data are generated as follows:

1. Send N_1 ($N_1 = 20$ in this case) requests to the safe and vulnerable web application, 10 for each,

2. Record the response time of each request.

A plot of data is given in Fig. 1 which shows network delays exist in both safe and vulnerable and varied in each request. This indicates the stochastic nature of the network delay. The statistics of the data are given in Tab. 1 from which we see the statistics of network delays in requesting the two urls are close. This means the network delays are independent of the types of web application we request. Now, we can assume the network work delay follows a certain probability distribution.

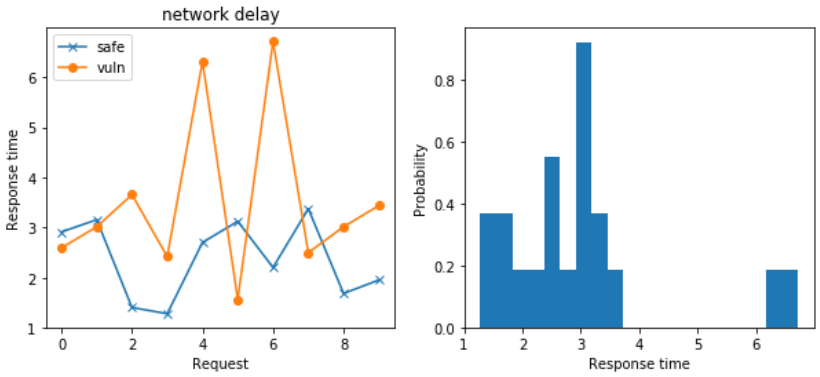

Figure 1: Network delay: response time (left), histogram of response time(right)

Table 1: statistics of the network delay data

url type	mean	standard deviation
safe	2.38	0.73
vulnerable	3.52	1.59

3 time-based algorithms

3.1 t test approach

Reason of using t test: To compare the models in (1) and (2), we rewrite them as

$$y_s = e \tag{3}$$
$$y_v = e + \theta x \tag{4}$$

where y_s, y_v denote the response times of safe and vulnerable urls, respectively, and e is a random variable denote the network delay, x is the sleep time (deterministic) given by the request, $x = 0$ for a normal request, θ is a scaling factor. Since the variance of a deterministic variable is 0, we can calculate the mean and variance of y_s and y_v as

$$\text{Mean}(y_s) = \text{Mean}(e) \tag{5}$$
$$\text{Mean}(y_v) = \text{Mean}(e) + \theta x \tag{6}$$
$$\text{Var}(y_s) = \text{Var}(y_v) = \text{Var}(e) \tag{7}$$

From (5)–(7), we see that y_s and y_v have same variances and different mean values (asymptotically or sample number $N \to \infty$) in the attack request where $x \neq 0$. Therefore, t **test is suitable for our case since it can determine whether the means of two sets of data are significantly different from each other or not**.

Actually, what we need to do is to decide whether response time (denoted by y) under the attack request has the equal mean with network delay(e). If they are the same, we can conclude that the url is safe, otherwise it is vulnerable. Therefore, the hypothesises of t test can be established as

- H_0: $\text{mean}(y) = \text{mean}(e) \iff$ Safe
- H_a: $\text{mean}(y) \neq \text{mean}(e)$ or $\text{mean}(y) > \text{mean}(e) \iff$ Volunerable

The detailed implementation of t test is conducted as follows:

1. send N ($N = 5$) requests to the url with sleep time= 2
2. store the response time in y
3. do the independent two-sample t test using the python function "stats.ttest_ind"

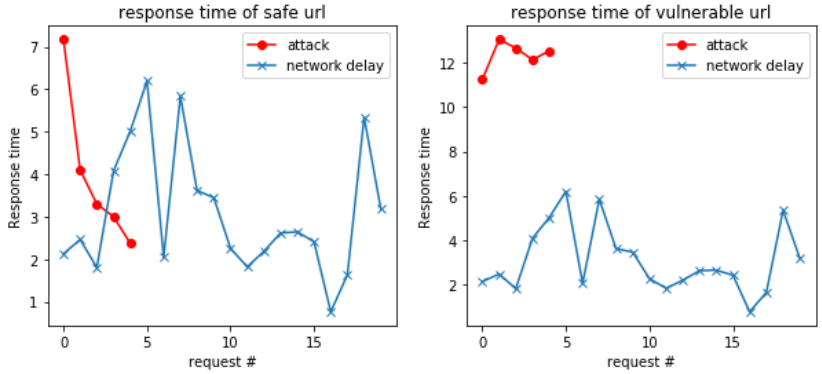

Figure 2: the plot of the response time vs the network delay

Figure 3: box plot comparison of the response time y and the network delay

4. calculate the p value and compare with the significance level $\alpha = 10^{-6}$

Note that the number of attack requests (e.g. $N = 5$) can be much less than the number of samples for the network delay (e.g. $N_1 = 20$). We do this for reducing the overall test time since the test time is dependent on N and the given sleep time. In this case, $N = 5$, the test time for the safe url while test time is around 15s and around 60s for vulnerable url. $N \geq 2$ for statistical validity.

Then the test is conducted. The response time of each request is presented in Fig.2 in red dots. The box plot in Fig.3 shows the mean value of response times is quit close to the network delay in the safe url (left of Fig.3) yet much larger than the mean of network delay in the vulnerable case (right of Fig.3).

If the results of t test show that the p value $p < \alpha$, we reject H_0 and

conclude the url is vulnerable, otherwise, the url is safe. In this test, the p values are given in Tab.2 which correctly identify the vulnerability of the urls with false positive rate less than $\alpha = 10^{-6}$. Note that we decrease alpha to guarantee a low false positive rate which, on the contrary, will cause the increase of false negative rates (i.e. fail to detect the vulnerable urls). We should pay attention to this fact. The detailed calculation of the t test is not given here for conciseness. The Python code for this approach is named "solution.py".

Table 2: results of t test

url type	p value	decision	test time
safe	0.35	accept $H_0 \Rightarrow$ safe	15.13s
vulnerable	2.97×10^{-7}	reject $H_0 \Rightarrow$ vulnerable	60.33s

3.2 χ^2 test approach: an one-request test

An alternative approach is to view model (3) and (4) as regression models. Therefore, the problem becomes a model selection problem. By observation, model (3) is nested in model (4), i.e. when $\theta = 0$, model (3) equals to model (4). Then we use (3) as a general model which can be rewritten as

$$y = \theta x + e \tag{8}$$

Our task is to test if $\theta = 0$ or not. Then, we can build the hypothesis test as

- H_0: $\theta = 0 \iff$ safe
- H_a: $\theta \neq 0 \iff$ vulnerable

Under the hypothesis H_0, y is independent of the sleep time x. We can use a χ^2 test for independence test after some operations.

However, in this report, we propose an alternative way to do χ^2 test which aims to do the test by sending out **only 1 attack request**. Since the network delay ($e \geq 0$) does not follow a normal distribution, we assume a log-normal distribution to the network delay e. Under the log-normal assumption and H_0, we have

$$\log(y) = \log(e) \sim \mathcal{N}(\mu, \sigma^2) \tag{9}$$

where we can calculate the estimate $\hat{\mu}$ and $\hat{\sigma}^2$ from samples of network delay. Let $\epsilon = \log(y) - \mu$, we have

$$\epsilon \sim N(0, \sigma^2)$$

We can conduct the correlation test between ϵ and x which can be turned in to a χ^2 test as

$$\frac{N}{\sigma^2}\epsilon^T X(X^T X)^{-1} X^T \epsilon \sim \chi^2(1) \tag{10}$$

where X are the given values of sleep time x, $\chi^2(1)$ is the χ^2 distribution with 1 degree of freedom. The mathematical operations for deriving (10) are not given here for conciseness.

The implementation of the χ^2 test can be done as follows:

1. send a attack request to the url with a given sleep time, e.g. $x_1 = 3$, and record the response time as y_1.

2. Let $X = [\mathbf{0}_{N-1}, x_1]$ and $y = [Y_{N-1}, y_1]$, where $\mathbf{0}_{N-1}$ denotes a zero vector with $N-1$ elements, Y_{N-1} is $N-1$ response times taken from the samples of network delay, $N = 10$.

3. calculate the p value of the $\chi^2(1)$ test using (10)

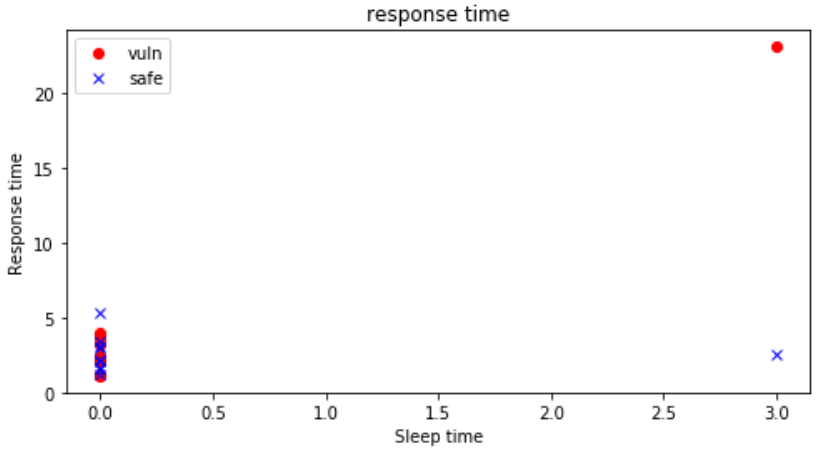

Figure 4: the response time y vs sleep time for the safe and vulnerable urls

The response time is given in Fig.4 from which we can see a clear dependency between the sleep time (x-axis) and the response time of the vulnerable url. The test results are presented in Tab. 3. The code of this approach is given in "solution2.py". The advantage of this approach is that the test time is much less than the t test approach. In this case, the test time is around 3s for the safe url and 20s for the vulnerable url. The test time can be further reduced by setting x_1 to a smaller number which is at the risk of increasing the type II error (false negative).

Table 3: results of χ^2 test

url type	p value	decision	test time
safe	0.65	accept $H_0 \Rightarrow$ safe	2.58s
vulnerable	5.5×10^{-16}	accept $H_0 \Rightarrow$ vulnerable	23.07s

4 Conclusions

Two time-based approaches are design for testing the vulnerability of web application. Both of them are effective and can be completed with in an acceptable time limit. The second approach is advantageous in test time (within a few seconds) since we only need to **send out 1 attack request**. This is beneficial when dealing with thousands of urls in real applications. In addition, numerical studies show that the χ^2 has much lower false negative rate. However, we pay the penalty of assuming that the network delay is log-normal distributed. If the assumption is strongly violated, the result may not be valid.

In addition, we mainly consider the type I error in this report. The type II error is also very important. To lower the type II error (false negative), we can set another hypothesis test to support our decision, e.g.

- H_0: $\text{mean}(y - e) = \theta x \iff$ Vulnerable
- H_1: $\text{mean}(y - e) < \theta x$ or $\text{mean}(y - e) \neq \theta x \iff$ Safe.

where θ can be calculated using samples. This test can be combined with the previous test results to detect the vulnerability of the url.

There are many other approaches can be developed. Due to the time limit, I only present the above approaches. The code for the two approaches are

stored in “solution.py” and “solution2.py”, respectively. They are readable and well documented.