Security Analysis of Bioinformatics WEB Application

Tao Tao[1], Yuan Chen[2(✉)], Bijing Liu[3], Xueqi Jin[4], Mingyuan Yan[5], and Shouling Ji[2,6]

[1] State Grid Hangzhou Power Supply Company, Hangzhou, China
taotao980925@aliyun.com
[2] Zhejiang University, Hangzhou, China
{chenyuan, sji}@zju.edu.cn
[3] NARI Group Corporation, Beijing, China
[4] State Grid Zhejiang Electric Power Co., Ltd., Hangzhou, China
[5] University of North Georgia, Dahlonega, Georgia
[6] Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies, Hangzhou, China

Abstract. Bioinformatics is a subject that focuses on developing methods and software tools, especially web applications, to analyze, understand and utilize biological data. This scientific field attracts large research interest and has been developed rapidly in most aspects but not on security. The lack of security awareness of researchers and insufficient maintenance are the main reasons for security vulnerabilities of bioinformatics web application, such as SQL injection, XSS and file leakage, etc. In the paper, we perform security analysis for website URLs extracted from PubMed abstracts, which contains more than 20,000 URLs. The analysis includes server version CVE matching, HTTPS security evaluation, git leakage detection, and small-scale manual penetration testing. The result shows that the most commonly used server version is outdated and vulnerable. Particularly, only one-fourth HTTPS domains are secure based on our testing, which only count for 7.6% in the entire testing websites. Discovered vulnerabilities are reported to website manager by email and we receive positive feedbacks.

Keywords: Bioinformatics · WEB Security · Git leakage

1 Introduction

In this section, we first introduce the background includes Bioinformatics, data source PubMed, WEB Security and some web security issues. After that, the motivation and problem under investigation are presented.

1.1 Bioinformatics WEB Application

Bioinformatics is an interdisciplinary field of biology, computer science, mathematics, information engineering, and statistics [1]. The research focuses on analyzing and interpreting biological data, in which computer programming plays a significant role.

C.-N. Yang et al. (Eds.): SICBS 2018, AISC 895, pp. 383–397, 2020.
https://doi.org/10.1007/978-3-030-16946-6_30

For example, The Basic Local Alignment Search Tool (BLAST) [2] is a sequence similarity search program, which can be used for inferring functional and evolutionary relationships between sequences as well as helping identify gene families.

In pursuit of better user experience and usability, many researchers develop a web interface. Users can use these applications thorough browser to utilize the software or query the database, so that save time and resources by avoiding downloading, compiling and installation. Take BLAST as an example, NCBI provides a public interface at https://www.ncbi.nlm.nih.gov/BLAST/, which has been cited by more than a thousand papers.

1.2 PubMed

From where can we get bioinformatics research papers? PubMed provides an answer. It includes more than 28 million citations for biomedical literatures. XML format data can be retrieved from NCBI FTP[1]. Though no full text is available on PubMed, it's universal for authors to provide a URL link to their website in paper abstract if their paper is releasing a new bioinformatics database or software. Therefore, we are interested in extracting URLs from PubMed abstracts, and perform security analysis on these websites.

1.3 Web Security of Bioinformatics Applications

Most web applications contain severe security vulnerabilities, which allow hackers access, manipulate data or even make remote code execution without authorization. Even for experienced programmers, identifying potential bugs and security vulnerabilities is a challenging task, let alone for biological researchers.

Furthermore, the develop-publish-graduate pattern in this field also make it worse for bioinformatics web application security. These applications become unmaintained after the developer which typically students graduate. As more and more software vulnerabilities expose nowadays, it is unquestionably not safe to stick to an old version. Even if the code which web application developers write is secure, the underlying software and operating system may be outdated and vulnerable for n-day attacks.

Based on these observations, we speculate that the security status may be even worse for bioinformatics web applications.

1.4 Web Security Issues

SQL Injection

If user provided data concatenates into a SQL query without sanitization, SQL injection occurs. By exploiting this vulnerability, attackers can do arbitrary operations like query, insert, or delete data in the database, leading to data get stolen, lost, corrupted, altered, even the server may be taken over.

[1] ftp://ftp.ncbi.nlm.nih.gov/pubmed/baseline.

Command Injection
Just like SQL injection, command injection occurs when unsafe user-supplied data being passed to a system shell command. Command injection is much more sensitive for web applications, because it allows arbitrary file read which can be leveraged to achieve database connection secrets leakage. Other sites running on the same host will also be affected if no proper isolation or access control made.

Cross Site Scripting (XSS)
Different from server-side code execution, the attack code of Cross Site Scripting (XSS) runs on users' browsers. According to HackerOne, in 2017, XSS was the most common attack type discovered by hackers [3]. Stealing user login credentials, making requests without user approval, and installing malicious software on user's computer are made viable by XSS.

XSS can be classified into two types, reflected XSS and stored XSS. These two types can be differentiated by checking whether server stores attack vector. In reflected XSS, victim clicks attacker crafted URL resulting in the execution of the malicious script embedded in the URL. This type of URL is easy to detect and most recent browsers provide protection techniques to make it much harder to exploit. In stored XSS, also known as persistent XSS, the attacker's payload is stored on the server. And all visiting users will run the payload. For example, a job submitting form allow submitting a job title and description, which are also shown in the result page. Stored XSS occurs when no adequate sanitization performed before outputting the page, affecting all users who visit this page.

1.5 Motivation and Objective

Find existing vulnerabilities and potential security risk inside a web application is essential. Although research proposed web applications may not implement a user system, neither process sensitive data like financial or e-commerce applications, which makes attacks like XSS less profitable, it's still a significant issue affecting the reputation of research institute once being attacked. More importantly, it is possible that these vulnerable applications being used or further developed to deal with sensitive human-related data, like medicine or health advisory service. Discovering vulnerabilities of bioinformatics web application and fix them is meaningful and necessary. Unfortunately, no existing literature on this topic can be found.

Therefore, this work tries to figure out whether or not bioinformatics web application is secure, and answering the following questions:

- Are these websites still available after paper publication?
- What and which type of server they are running?
- Are these server program outdated or with known CVEs?
- How many of them deploy HTTPS?
- Is HTTPS deployed securely or vulnerable for existing attacks?
- Are these web applications vulnerable to typical web security issues?

The contribution of this work can be summarized as follows:

- As far as our knowledge, we are the first to explore the current security situation of bioinformatics web applications.
- Our tests towards more than 20,000 URLs proved that most bioinformatics web applications are running outdated and vulnerable server version, and HTTPS deployment is far from satisfaction.
- We conducted git leakage detection and small-scale manual penetration test, reported discovered vulnerabilities to contact email addresses, and received positive feedbacks.

The remaining contents are organized as follows. In Sect. 2, we present the used method in detail. Our findings and discussion are given in Sect. 3. Section 3.6 shows a case study as a realistic attack example, from git leak to root privilege. Then we summarize and propose future work in Sect. 4.

2 Method

In this section, we will introduce the whole procedure of our analysis. The source of our data will be described in Sect. 2.1, then we perform accessibility analysis, server version detection, HTTPS security rating and git leakage in the following sections. Section 2.7 ends with reporting discovered security issues to contact email addresses.

2.1 Data Source

DaTo [4] is a database of biological software and database. It extracts URL from abstracts from all PubMed papers and uses text mining to locate the tool name. Then, it leverages E-link API provided by NCBI to fetch the citation list between papers, to provide a user-friendly interface for search, map, statistics and network functions. Figure 1 shows the workflow of DaTo.

This work uses DaTo dataset updated in 2017 July. One of us also committed to the development of DaTo, so we can access the dataset. We believe DaTo covers most bioinformatics web applications and outperforms other datasets. DaTo provides 19 data fields, including `pmid`, `lat`, `lng`, `name`, `description`, `url`, `urlstatus`, `country`, `has_mesh`, `abstract`, `date`, `mesh_term`, `jid`, `journal_short`, `country_long`, `state`, `city`. In these data fields, pmid, url, urlstatus, and date are used in this work. Pmid means PubMed ID, url is the URL extracted from the paper abstract, urlstatus shows whether this website can be accessed or not. The date shows the publication time of a paper.

DaTo dataset comprised of 23452 possible bioinformatics tools and databases. After inspecting the data, we found that some URL items are paper publication URL, DOI URL, copyright statement or FTP addresses. Since these URLs are not related to bioinformatics web application, so we excluded these data items from the dataset. After this pre-processing step, 22786 entries retained.

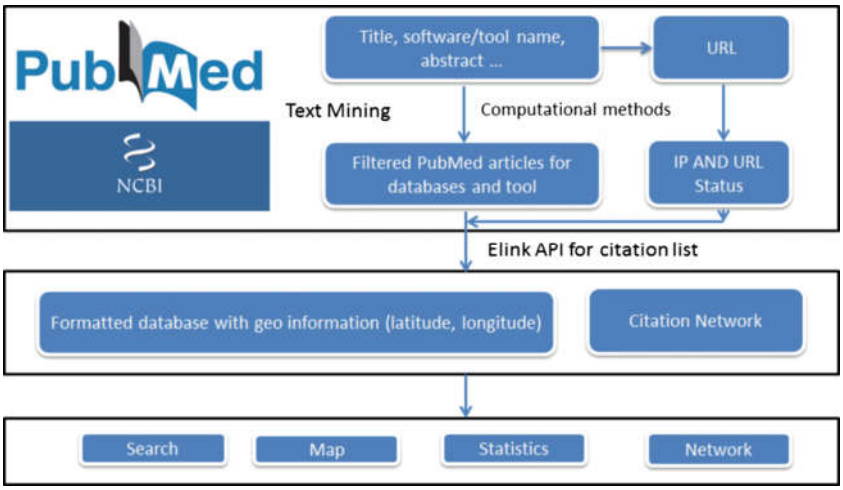

Fig. 1. Data processing workflow of DaTo (From [4])

2.2 Accessibility Analysis

If a web application is not accessible because the server is down, domain is expired, or return 404 errors, its security is out of the question. So, before performing any security analysis, we need to determine whether these applications are still available and online.

DNS Resolution. When a user visits a website, DNS (Domain Name System) provides translation from domain to IP address. If the DNS resolution failed to return an IP address, most likely the domain has expired, and the website is unquestionably not accessible. In this step, we query system default DNS server for A-type DNS response of these domain names. If `NXDOMAIN`, `NoNameservers`, `Timeout` or any other errors occurred more than three times, we marked it as a failure.

Ping Test. Ping is a simple tool using ICMP network protocol for determining whether the host is online. Packet loss rate and network delay can be obtained by pinging target host. In this step, IP addresses collected from domain name resolution are being tested using Ping, we mark it as a failure if it fails to respond three times.

2.3 Server Software Security Analysis

Server Version Retrieval

In HTTP protocol, the server version information is included in the HTTP response message `Server` header field. Censys.io [5] is a search engine for network host, which has collected more than three billion ipv4 addresses using Zmap.

In this step, we query IP information from Censys.io API to get the web server information. Here is an example:

```
131.204.46.201 Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/
5.6.30
```

As it appears, we can learn that the target host is running on a CentOS operating system, using Apache HTTP Server 2.4.6, OpenSSL 1.0.2k, and PHP 5.6.30.

CVE Vulnerabilities Retrieval
OWASP (Open Web Application Security Project) Top Ten aims to raise awareness about application security by identifying some of the most critical risks facing organizations [9]. Using Component with Known Vulnerabilities is one of the major concerns listed. Thereby it's essential to match software version with known vulnerabilities. CVE (Common Vulnerabilities and Exposures) provide a unique ID for public security issues, including ID, description and at least one public reference link. However, risk rate and fix suggestions are not included in the CVE official website, yet NVD provides these data.

cvedetails.com provides an easy-to-use web interface to query CVE entries for a specific vendor, software or specific program version. By combining exploit data from exploit-db.com, cvedetails.com also provides public exploit information. It is helpful to determine whether a CVE is more severe so that get it fixed as soon as possible.

To automatically match software version with its corresponding CVE information, we program to crawl the data provided by cvedetails.com. Starting from searching software name, fetching its version list, and get CVE info for each version. CVE info obtained includes CVE ID, risk rating, description, and public exploits count.

2.4 HTTPS Support Test and Security Rating

HTTPS is an extension of the Hypertext Transfer Protocol (HTTP) for secure communication. It leverages TLS layer to provide a secure channel for HTTP. The HTTPS certificate is used to verify the identity of connection host, which makes MITM (Man in The Middle) attack much harder towards HTTPS connections compared with HTTP connections. Modern browsers also employ a stricter security policy for HTTPS sites, for example if an HTTPS website load insecure HTTP scripts, the browser will show a warning indicating insecurity.

HTTPS Support Test
This step is implemented in Python script, using Request library to try to make https connections. Using parameter `verify=False`, making TLS connection regardless of certificate errors, gives us a list of domains which support HTTPS connection while parameter `verify=True` is leveraged to filter websites deployed correct and trusted HTTPS certificates from the aforementioned list.

HTTPS Security Rating
SSL is a complex and hybrid protocol, allowing numerous features at different connection stages. Therefore, it is challenging to identify the HTTPS security level of a target website. To address this and make our test result more persuasive, we use HTTPS rating API provided by Qualys SSL Labs. Concretely, we use ssllabs-scan to obtain the rating result. The command is as follows:

```
./ssllabs-scan -usecache=true -verbosity=debug --grade -hostfile https_verified.txt > ssltest_output.txt
```

The procedure of generating HTTPS security rating proposed by Qualys SSL Labs HTTPS is as follows:

1. Checking whether the certificate is valid and trusted. If not, give grade M for Certificate Mismatch or grade T for Certificate Untrusted and skip the following checks.
2. Producing a weighted centesimal score based on three dimensions: supported protocols – 30%, supported key exchange methods – 30%, and supported encryption methods – 40%. In particular, if a score for one dimension is zero, the total score is set to zero.
3. Using the following rule to transform the score to grade A to F:
   A: [80,100], B: [65,80), C: [50,65), D: [35,50), E: [20,35), F: [0,20)
4. Some special rules may be applied. Such as reducing from A to A- if unwanted features are present, and using A+ for unusual secure configuration.

We refer [6] for detailed evaluation standard of the rating process.

2.5 Git Leakage Detection

Git is a famous version control tool, especially for managing source code. Some web applications may leverage Git to control the version, such as using webhook to automatically pull the latest code files, so that developers can get rid of trivial work like copy files to the server. But, Git has its drawback if HTTP server wrongly configured, which is Git leakage.

Git leakage means that the .git folder is accessible via GET HTTP request. Attackers can download the entire or partial .git folder using the knowledge of Git internals like the structure of .git/index and zlib compression. Attackers can use tools like GitHack[2] to conduct an attack.

For simple detection propose, we only need to access .git/HEAD. If the server responds with code 200 in the HTTP response and the file content match the desired file format, we consider it has a Git leakage vulnerability.

Although Git leakage will make the source code public, there is no actual harm for open-source websites. In order to exclude open-source websites, we filter those public repositories. We fetch the branch ref file, such as .git/refs/heads/master for branch master (typically listed in .git/HEAD file), to get the latest commit ID. Then we query GitHub API based on the commit ID to filter those open-source websites. The remaining domains will be kept for further vulnerability notification.

2.6 Manual Penetration Test

Although there are numerous existing automatically or semi-automatically vulnerability black-box scanners, we choose not to use these tools due to ethical and legal consideration. Using scanners may put tremendous pressure on the target server, or

[2] https://github.com/lijiejie/GitHack.

even cause a deny of service attack. So, in this step, we choose a small number of websites to test their common vulnerabilities from the following aspects manually:

- Dangerous File Upload: Try to upload a harmless file such as phpinfo.php through a file uploading form, and guess the uploaded location.
- Sensitive File Leakage: For a result page, access the folder page to check if directory listing is enabled. For a dynamic file read page, such as with parameter `?file=job_id.txt`, change the parameter to check if system file `/etc/passwd` can be accessed.
- XSS Attack: Add ">>>" to GET parameters, to check if ">>>" is embedded in the response page. If only ">>>" is present, the website has taken defense against XSS into consideration.

Notably, for websites providing source code download and written in PHP, we use RIPS-0.55 to audit the source code. RIPS is an open-source PHP static analysis software to detect security vulnerabilities such as XSS, SQL Injection, Local File Inclusion, and others.

2.7 Vulnerability Report

After found and confirmed the vulnerability, we decide to report the problem we found to the website owner. According to paper [7], e-mail notifications is the best and economical method to send the alert letter.

We collect the contact email from the web pages manually, such as from the bottom of the index page or contact page, if not present, we seek to use the email address of the paper corresponding author.

"WEB Security issue about {{domain}}" is used as email subject, and in the email body, we provided necessary information about the vulnerability we found such as vulnerability type, url and impact, besides we also provided fix suggestion and reference urls.

3 Result and Discussion

3.1 Accessibility Analysis

After data cleaning, there are 22786 URL data items in DaTo dataset, which contains 12276 domains and 86 IP addresses. DNS lookup found that 1619 (13.2%) domains cannot resolve, which counts for 2214 (9.7%) URLs.

The length of IP list from DNS lookup plus IP address is 8010.

Moreover, the Ping test showed that only 4063 (50.7%) IP replied to our Ping request. Though this does not mean only half hosts were online at that moment, because some hosts may be configured not to answer ICMP messages.

3.2 Server Software Security Analysis

The query from Censys.io returned 5891 data items, in which 3500 (59.4%) successfully matched to known software version. The primary reasons of mismatching are rare software which has no CVE information, and WAF or CDN deployed which hide the Server field.

Figure 2 and Table 1 shows the most frequent Server version and matched CVEs.

Table 1. Top 10 frequent server software version and corresponding CVEs counts

Server software version	Counts	CVE counts	High-risk CVE counts	Public exploits count
Apache/2.2.15	480	28	6	2
Apache/2.4.6	285	21	1	1
Apache/2.2.22	277	15	5	0
Apache/2.4.7	266	20	1	1
Apache/2.2.3	242	50	10	2
Microsoft-IIS/7.5	174	5	3	1
Apache/2.4.10	172	17	4	0
Apache/2.4.18	168	14	4	0
OpenSSL/1.0.1e	154	68	13	2
Apache-Coyote/1.1	149	0	0	0

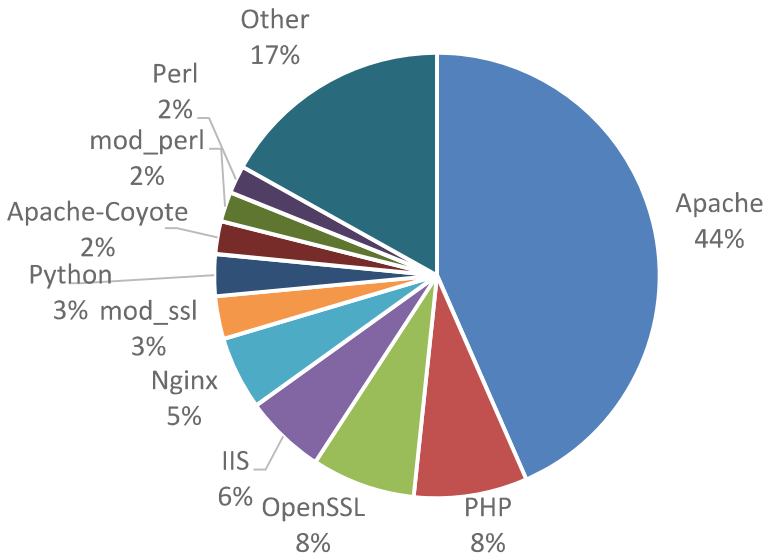

Fig. 2. Frequency distribution of server software

Here we define those CVEs whose CVSS score greater than 7 are high-risk CVEs. Public exploits are exploits that are uploaded to exploit-db.com to exploit the CVE. As we can see from this result, it's evident that a large proportion of servers are running outdated server version software, and there're public exploits attackers can make use of.

In 5891 IP addresses, 3109 (52.8%) of them successfully matched to CVE entries, in which 2991 (50.8%) have at least one high-risk CVE, and 2028 (34.4%) have at least one CVE that corresponding public exploit is available.

It is necessary to note that even high-risk CVEs may not be easy to make use. A specific vulnerability only exists under some particular requirements and environments. Take heart-bleed (CVE-2014-0160) as an example, this vulnerability can lead to leakage of users' cookies and passwords. Yet, if the server has not configured HTTPS, this CVE has no impact to this server, so attackers cannot exploit this vulnerability. Besides, the public exploit may merely be a POC (Proof of Concept), and there is a considerable gap between CVE and real impact. Nevertheless, more CVEs indicate less security and more threats.

Figure 3 shows the geographical distribution of the average count of public exploits per country. The darker, the problem is more severe. We can see developed countries like the United States and European countries are less vulnerable, by contrast, developing countries are more likely to use the vulnerable software.

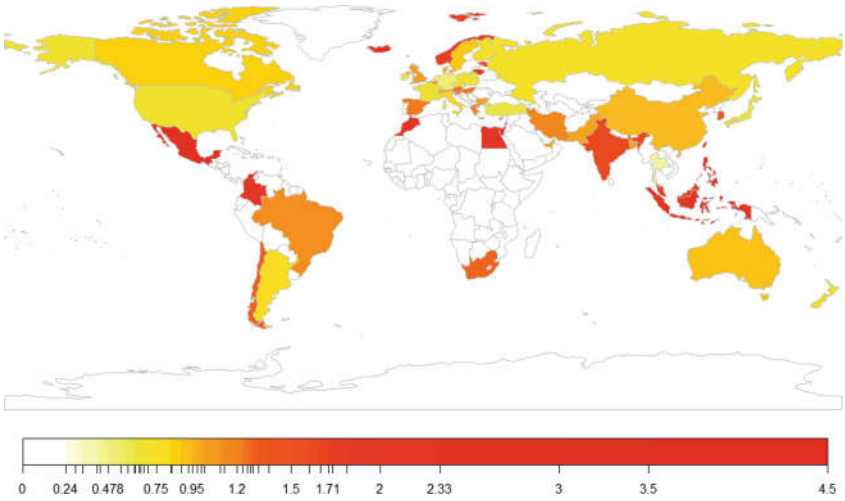

Fig. 3. Geographical distribution of vulnerable bioinformatics web applications. The value is average number of public exploits per country.

Possible explanations for this phenomenon may include the difference introduced by economic level. Economically advanced countries are also technology developed countries, so their developers' security awareness is higher than those in developing countries. Besides, newer applications developed also contribute to less vulnerable applications exploitable.

3.3 HTTPS Support Test and Security Rating

After DNS lookup, we have 10687 resolvable domains and IPs. HTTPS connection without verifying certificate proved that 5278 (49.4%) of them support HTTPS connection, which takes about half proportion. Meanwhile, supporting HTTPS and having a valid and trusted HTTPS certificate only accounts for only one fourth, which are 2727 (25.5%). This data shows that in this field, HTTPS is far from being adopted extensively.

Calling security rating API by ssllab-scan provided by Quals SSL Lab to test the aforementioned 2727 domains/IPs, we collected 1346 data items. The main reasons for missing data are network fluctuation and target server rejection of being tested. We plot the distribution of returned grade data in Fig. 1. In which A and A+ applications count for 60.3% (811), which is a desirable result. However, they only account for 7.6% of entire domains/IPs.

To sum up, only one-fourth of bioinformatics web applications deployed valid HTTPS certificate, and only 7.6% applications deployed secure HTTPS configuration (Fig. 4).

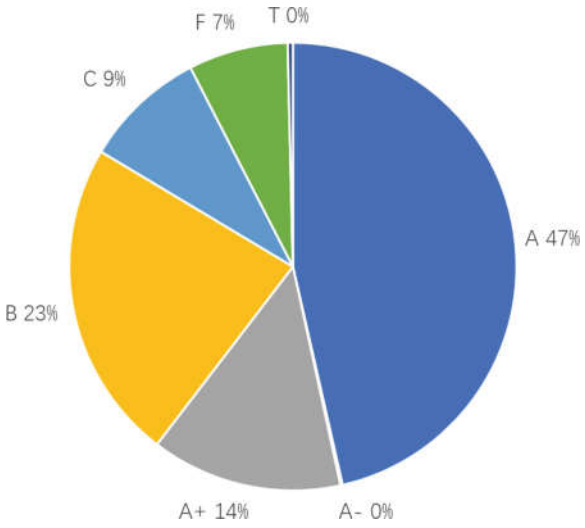

Fig. 4. HTTPS security grade distribution

3.4 Git Leakage Detection and Manual Penetration Test Result

From 18236 distinct URLs, we access each one's corresponding .git/HEAD file, and we found 134 (0.73%) of them have git leakage issue. After querying commit ID of branch ref file, we filtered some open-source projects to get a list of 91 URLs. Then, we manually visited each one to get the contact email addresses and ignore those static websites (only providing description or software download functionality). After this step, we have 55 contact email addresses to send the notification letter.

Manual penetration test result is limited to our technical level and ethical consideration. Nevertheless, we still get some insights proving the existence of typical types of WEB vulnerabilities. Table 2 summarize our findings of manual test.

Table 2. Findings of manual penetration test

Vulnerability type	Count
Cross Site Scripting (XSS)	9
Dangerous file upload	2
Remote command execution	1
Arbitrary file reading	1
Directory listing	1

3.5 Vulnerability Report

We send email to 66 email addresses of 55 websites in sum. If rejection letter received, we try to find another possible email address to send. Then, we got nine replies in total, in which 4 of them finished the repair, one finished part of the work but still need further amendment, one responder said he would consider it (XSS defense) in new version development, and two replied saying not a security threat due to open-source (Git leak issue).

All replies are positive about our work and expressed gratitude to us. Here is a letter sample:

```
Dear ***(redacted),
  thank you very much for pointing out this issue.
  For safety reason, we should deal with it as soon as
possible. I will contact the current Website maintainer
and server administrator to figure out a solution.
  I appreciate your work in Bioinformatics Security and
active involvement in improving security levels. As a
very inexperienced Web-developer by then (even until
now), I often ignored this aspect as long as the code
worked. I hope starting from this issue I will learn more
knowledge about Web security and contribute to building a
better bioinformatical environment.
  Thank you again!
Best regards,
***(redacted)
```

3.6 Case Study – from Git Leak to Root Privilege

In this section, we will show how a bioinformatics web application can be compromised to get root privilege of the target host. We denote the target website as `vuln.com`, denote username as `user`, project name as `name`, our server IP as `our_ip`, and mask some data fields for ethical consideration.

First, from the result of Git leak scan, we found http://vuln.com/.git/config is present, which tells us the remote origin is `git@gitlab.com:user/name-server.git`. By visiting corresponding GitLab project page, we found this website source code is open-source under MIT license.

The repository has more than two hundred commits and contains more than 200 MB files. Most files are committed three years ago (2015), and the last commit is authored in October 2017. Besides, this repository has no README file nor informative description text and has 0 stars. So, this repository can be viewed as not in actively developing and less-concerned.

By searching dangerous functions like shell_exec and exec, we found numerous calling to these dangerous functions with $PARAM. This coding style is rather dangerous, just like concatenating variables to SQL statements, if no sanitization is assured before the function call for any involved variable. Here is a code example extracted from `align.php`:

```
$results_dir = $_REQUEST['results_dir'];
$pairwise = shell_exec("cd $results_dir;echo -n `grep
PAIRWISE $PAR_FILE`");
```

As you can see, the variable results_dir is directly passed to shell_exec calling. So, the attacker can easily exploit this remote code execution vulnerability, just like visit this URL below:

```
http://vuln.com/align.php?results_dir=|whoami#
```

But here the command execution result is not shown in the response page, so we need to send the execution result to our server. Here is the modified proof of concept:

```
http://vuln.com/align.php?results_dir=|curl our_ip:6666
--data `id|base64 -w0`#
```

This exploit leverage the curl utility which typically embedded in the system to send the base64 encoded command output to our server via HTTP POST body. (Our server is running `nc -lp 6666` to receive data). The base64 encoding is introduced to avoid interference of special characters in the command output (Fig. 5).

```
root@MyServer:~# nc -lp 6666
POST / HTTP/1.1
User-Agent: curl/7.35.0
Host: [redacted]:6666
Accept: */*
Content-Length: 112
Content-Type: application/x-www-form-urlencoded

dW1kPTMzKHd3dylkYXRhKSBnaWQ9MzMod3d3LWRhdGEpIGdyb3Vwcz0zMyh3d3ctZGF0YSksOTk
```

Fig. 5. Our server successfully receives the command execution result from the vulnerable host.

Base64-decoding the string gives us: (partially marked):

```
uid=33(www-data) gid=33(www-data) groups=33(www-data),999
(gitlab-www),1000(***)
```

So, attackers can do anything from reading system files, to delete all files owned by www-data. But this is not the end, just like matching HTTP Server version to CVE, we can also check the Linux kernel version to find whether privilege escalation vulnerabilities are present. So, we do cat/etc/issue, and uname -a to get the knowledge of the system. And the corresponding results are:

```
Ubuntu 14.04.5 LTS \n \l
Linux *** 3.13.0-73-generic #116-Ubuntu SMP Fri Dec 4
15:31:30 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
```

It is clear that stick to an old version kernel is definitely not safe. This kernel version is vulnerable to Dirty COW (CVE-2016-5195). Linux kernel 2.x through 4.x before 4.8.3 are all affected by the Dirty COW [8], leaving attacker an easy way to gain root privilege because of public exploit script (https://www.exploit-db.com/exploits/40839/).

Due to ethical considerations, we did not perform any escalation attempt so that we cannot be certain about root privilege acquisition, it is possible that maybe this server has been patched for this CVE. But, by searching CVE list for this kernel version, there are many more CVEs which malicious attacker can use. (cvedetails.com has 174 CVEs for kernel version 3.13.) So it is definitely a severe threat to the whole system that needs to be fixed as soon as possible.

4 Conclusion and Future Work

In this paper, we evaluated bioinformatics web applications' security from multiple dimensions and found several worrisome facts, such as three-fourths of them still doesn't support HTTPS, and about half of them are using server software with high-risk CVEs, and even 34.4% of them are at risk of public exploit scripts.

After git leakage detection and manual penetration test, we reported security issues to website owners and corresponding authors and received positive responses. This proved our inspection of problem reason, which is the developers' inadequacy of security awareness.

Bioinformatics security is a research field that far from being studied. As fast development of bioinformatics and medicine, more and more related web application or APPs will be developed and commercialized. Security and privacy are critical requirements for these advancements and should be addressed before it's too late. How to better apply security scanner to bioinformatics area, how to get authorization from website owner and how to do a better vulnerability notification campaign are left to future work.

Acknowledgement. We would like to thank the anonymous reviewers for their valuable suggestions for improving this paper. We are also grateful to Yincong Zhou, Dahui Hu and Prof. Ming Chen of The Group of Bioinformatics of Zhejiang University for their work about DaTo and contribution to this work.

This work was partly supported by NSFC under No. 61772466, the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under No. R19F020013, the Provincial Key Research and Development Program of Zhejiang, China under No. 2017C01055, the Fundamental Research Funds for the Central Universities, and the Alibaba-ZJU Joint Research Institute of Frontier Technologies. Technology Project of State Grid Zhejiang Electric Power co. LTD under NO. 5211HZ17000J.

References

1. Bioinformatics Wikipedia. https://en.wikipedia.org/wiki/Bioinformatics. Accessed 12 Oct 2018
2. Johnson, M., et al.: NCBI BLAST: a better web interface. Nucleic Acids Res. **36**(2), W5–W9 (2008)
3. Ranger, S. At $30,000 for a flaw, bug bounties are big and getting bigger – ZDNet. http://www.zdnet.com/article/at-30000-for-a-flaw-bug-bounties-are-big-and-getting-bigger/. Accessed 12 Oct 2018
4. Li, Q., Zhou, Y., et al.: DaTo: an atlas of biological databases and tools. J. Integr. Bioinform. **13**(4), 297 (2016)
5. About Us – Censys. https://censys.io/about. Accessed 12 Oct 2018
6. SSL Server Rating Guide. https://github.com/ssllabs/research/wiki/SSL-Server-Rating-Guide. Accessed 12 Oct 2018
7. Stock, B., Pellegrino, G., Li, F., et al.: Didn't you hear me? - towards more successful web vulnerability notifications. In: Network and Distributed System Security Symposium (2018)
8. CVE-2016-5195 in Ubuntu. https://people.canonical.com/~ubuntu-security/cve/2016/CVE-2016-5195.html. Accessed 12 Oct 2018
9. OWASP Wikipedia. https://en.wikipedia.org/wiki/OWASP. Accessed 12 Oct 2018