同济大学计算机系

信息安全原理 PROJ3 报告

学　　号　2152701

姓　　名　陈玟桦

专　　业　信息安全

授课老师　谭成翔

一、实验内容

1.1 需求分析

本次实验的需求主要分为两类：

（1） 查询资料，了解 OpenSSL 的漏洞类别和特征，以及漏洞检测程序的原理

（2） 在 Linux 下实现一个基本的漏洞检测系统

1.2 实验环境

Ubuntu20.04 版本下 python 以及 c++实现

二、OpenSSL 漏洞类别及特征

2.1 计时攻击缺陷

在针对对称加密算法和非对称加密算法计时攻击的研究中，目前主要是结合高速缓冲存储器 Cache 的行为信息作为破解密钥的思想。Cache 计时攻击的本质是密码加解密过程中微处理器读取 Cache 数据时会发生 Cache 命中或者 Cache 失效从而导致存在时间差异，结合算法设计结构推测其内部执行过程，实现密钥破解。

2.2 分支预测缺陷

现代处理器采用分支预测机制，即在分支指令执行结束之前利用预测算法猜测哪条分支将会被运行，以提高处理器指

令流水线的性能。然而，利用来集度量分支被正确预测和错误预测时不同的执行时间，就可以分析出处理器执行了哪一条分支。在某种意义上，分支预测攻击也是计时攻击的一种。

2.3 故障分析缺陷

OpenSSL 在针对对称和非对称加密算法的具体实现上存在缺陷，易遭受故障分析攻击威励。故障分析攻击的思想最早由 Boneh 等人提出，该攻击利用密码运算过程中的错误信息实施攻击。之后，文献提出应用于对称密钥算法的差分故障分析，成功破解 DES 算法密钥。

2.4 伪随机数生成器缺陷

随机数在密码学中起着关键作用，很多密码算法都要将随机数作为基础参数来运用，例如，直接以随机数作为公钥密码算法中的密钥，或是以随机数来产生非对称加密算法中的密钥或对称加密算法中的会话密钥:在密钥分配中，使用随机数来防止重放攻击等。

2.5 拒绝服务缺陷

DoS 是 DenialofService 的简称，即拒绝服务，造成 DoS 的攻击行为被称为 DoS 攻击。虽然具体的实现方式千变万化，但都有一个共同点，即其根本目的是用超出自标处理能力的海量数据包消耗可用的系统资源、带宽资源等，使受害

主机或网络无法及时接收并处理外界请求，或无法及时回应外界请求。在 OpenSSL 针对 SSL/ CD) TLS 协议的具体实现中，存在着许多代码编写问题，如递归漏洞、空指针引用等。利用这些缺陷，很容易使客户端或者服务器产生拒绝服务。

2.6 Heartbleed 缺陷

在分析 Heartbleed 漏洞攻击之前，需要了解心跳协议。心跳协议，简单说就是数据通信中的节点需定期交换消息，即定期发送心跳信号，以确保远程数据服务中的节点能够检测到网络中断或节点崩溃。

(D)TLS 协议中的心跳扩展协议与普通网络使用的心跳协议类似，为了确保在使用(D)TLS 协议通信的双方，能够感知到对方的存在，保证其连接的有效性，引入了心跳扩展协议。然而，基于 OpenSSL 实现的心跳扩展协议存在边界检查漏洞，由于没有对缓冲区涉及的相关"长度"进行检查，准露的信息内容可能包含证书私钥、用户账户与密码、电子邮件以及重要的商业文档和通信等数据。

2.7 中间人攻击缺陷

中间人攻击 CMan-in-the-MiddleAttack(MITM)是一种由来已久的网络攻击手段，并且在今天仍然有着广泛的发展空间，例如 5MB 会话劫持、DNS 欺骗等攻击，都是典型的 MITM 攻击。儒而言之，所谓的 MITM 攻击就是在通信双方

毫无察觉的情况下，通过拦截正常的网络通信数据，进而对数据进行嗅探或篡改。OpenSSLSSLI(D)TLS 协议中实现的握手过程存在检查验证漏洞易被中间人利用，通过伪造修改密码规范数据或者证书达到中间人攻击的目的。

三、检测程序的原理

3.1 版本检测

首先，程序需要确定系统中安装的 OpenSSL 版本。这通常是通过执行 openssl version 命令或类似的系统命令来完成的，该命令会返回当前安装的 OpenSSL 版本号。例如，它可能返回类似于 OpenSSL 1.1.1d 的结果。

3.2 漏洞数据库

程序需要有一个已知漏洞的数据库，这个数据库包含了 OpenSSL 的历史版本及其已知的安全问题。这些信息通常可以从官方的安全公告、安全研究论文或者 CVE（Common Vulnerabilities and Exposures）数据库中获得。

3.3 版本比较和匹配

程序将检测到的 OpenSSL 版本与漏洞数据库中记录的版本进行比较。这通常涉及到字符串匹配或正则表达式匹配。如果检测到的版本与数据库中的任何一个已知漏洞版本

匹配，那么程序就会认定存在安全漏洞。

3.4 漏洞详情

一旦匹配成功，程序会查找并提供关于该漏洞的详细信息，包括漏洞的编号（如 CVE 编号）、影响的版本范围、漏洞的简要描述以及可能的修复建议。

3.5 输出和建议

程序会输出检测结果，如果发现漏洞，它会建议用户更新到一个安全的版本或者采取其他缓解措施。如果未发现漏洞，程序会通知用户当前版本是安全的。

四、Linux 下实现基本 OpenSSL 漏洞检测系统

本程序以 Python 爬虫读取公开漏洞数据库，将漏洞版本号存入 versions.txt 文件中，后续用 C++程序将匹配版本与本机 OpenSSL 版本进行比对的方式实现 Linux 下的 OpenSSL 漏洞检测系统，相当于利用了 3.1 中的检测原理，所使用的漏洞数据库是：

https://www.cvedetails.com/vulnerability-list/vendor_id-217/Openssl.html

完整源代码见 isp-03.py 以及 PROJ3.cpp 通过 Python 的 bs4 模块向目标网站发送服务器请求，并获取检测所需要的漏洞信息，这其中重要的工具函数如下：

```python
import requests
from bs4 import BeautifulSoup
import os

# 目标网页的URL
url = 'https://www.cvedetails.com/vulnerability-list/vendor_id-217/Openbsd/Openbsd.html'

headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36 Edg/123.0.0.0'}

# 发送GET请求，获取网页内容
response = requests.get(url,headers=headers)
web_content = response.text
# 用于存储txt文件链接的列表
txt_links = []
t_links = []

if response.status_code == 200:
    # 解析HTML内容
    soup = BeautifulSoup(web_content, 'html.parser')

    # 查找所有<a>标签
    links = soup.find_all('a', href=True)

    # 筛选出以/cve/开头的链接
    for link in links:
        href = link['href']
        if href.startswith('/cve/CVE-'):
            txt_links.append(href)
```

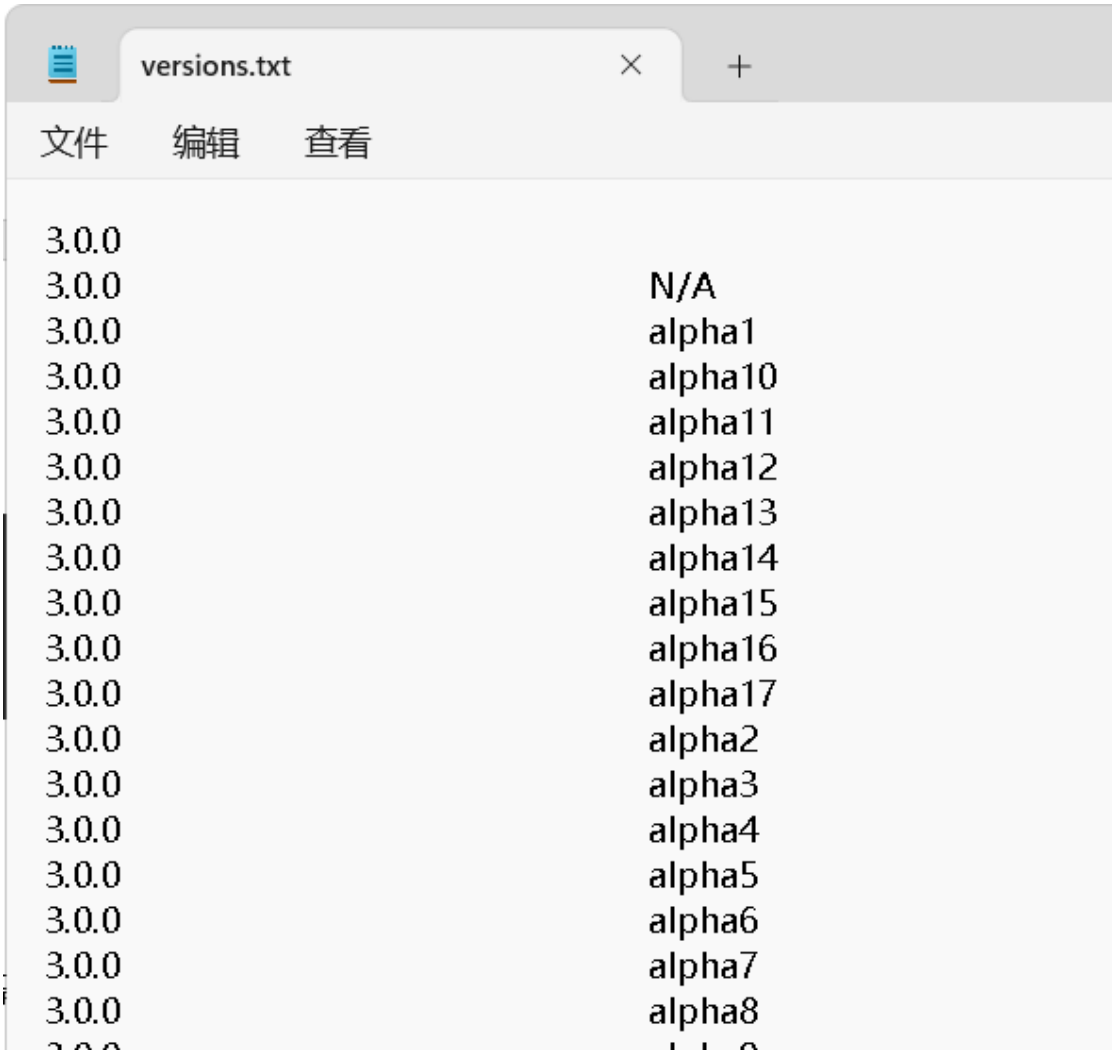
```python
# 下载并保存每个txt文件
for txt_link in txt_links:
    # 这里需要根据实际情况构造完整的文件下载URL
    file_url = 'https://www.cvedetails.com' + txt_link
    response = requests.get(file_url,headers=headers)
    if response.status_code == 200:
        soup = BeautifulSoup(response.text,'html.parser')

        Sec_links = soup.find_all('a',href=True)

        for Sec_link in Sec_links:
            Sec_href = Sec_link['href']
            if Sec_href.startswith('/version-search.php?cpeMatchCriteriaId='):
                t_links.append(Sec_href)

        for t_link in t_links:
            final_url = 'https://www.cvedetails.com' + t_link
            response = requests.get(final_url,headers=headers)
            if response.status_code == 200:

                soup = BeautifulSoup(response.text,'html.parser')

                tbody = soup.find('tbody')

                if tbody:
                    for tr in tbody.find_all('tr'):
                        fourth_td = tr.find_all('td')[3] if len(tr.find_all('td')) >= 4 else None
                        if fourth_td:
                            text = fourth_td.get_text(strip=True)
                            with open('versions.txt', 'a', encoding='utf-8') as file:
                                file.write(text + '\n')
```

之后，在 C++程序中我们获取 openssl 版本号将其与先前 python 程序获取的漏洞版本号进行比较，其中具体的代码实现如下：

```cpp
#include <iostream>
#include <string>
#include <unordered_map>
#include <regex>
#include <system_error>
#include <fstream>
#include <cstdlib>

// 执行shell命令并获取输出
std::string exec_command(const std::string& cmd) {
    std::array<char, 128> buffer;
    std::string result;
    std::unique_ptr<FILE, decltype(&pclose)> pipe(popen(cmd.c_str(), "r"), pclose);
    if (!pipe) {
        throw std::system_error(errno, std::system_category(), "popen failed!");
    }
    while (fgets(buffer.data(), buffer.size(), pipe.get()) != nullptr) {
        result += buffer.data();
    }
    return result;
}

std::vector<std::string> read_versions_from_file(const std::string& file_path) {
    std::vector<std::string> known_versions;
    std::ifstream file(file_path);
    std::string line;

    if (file.is_open()) {
        while (std::getline(file, line)) {
            // 去除尾部的换行符
            line.pop_back();
            // 去除中间多余的空格
            size_t start = line.find_first_not_of(" \t");
            if (start == std::string::npos)
                continue; // 忽略空行
            size_t end = line.find_last_not_of(" \t");
            line = line.substr(start, end - start + 1);
            known_versions.push_back(line);
        }
        file.close();
    } else {
        throw std::runtime_error("无法打开文件!");
    }

    return known_versions;
}

int main() {
    int count = 0;
    try {
        // 获取当前OpenSSL版本
        std::string openssl_version = exec_command("openssl version -a | grep 'OpenSSL' | awk '{print $2}'");
        std::cout << "当前OpenSSL版本: " << openssl_version << std::endl;
        // 从文件读取已知漏洞版本号列表
        std::vector<std::string> known_versions = read_versions_from_file("versions.txt");
        // 检测漏洞
        for (const auto& version : known_versions) {
            std::regex version_regex(version);
            if (std::regex_search(openssl_version, version_regex)) {
                std::cout << "警告: 发现已知漏洞! " << std::endl;
                count++;
            }
        }
        if(count!=0){
            std::cout << "该版本存在漏洞"<< count<<"个" << std::endl;
            return 1;
        }
        // 如果没有发现漏洞，输出安全信息
        std::cout << "当前OpenSSL版本是安全的。" << std::endl;
        return 0; // 没有漏洞，正常退出
```

```
71      } catch (const std::exception& e) {
72          std::cerr << "错误: " << e.what() << std::endl;
73          return 0; // 捕获到异常，退出程序
74      }
```

程序的运行结果可以分为两部分，一部分是爬虫程序爬下的漏洞版本号：



另一部分是检测程序的输出结果：

```
loli545799@ubuntu:~/Documents$ g++ -o PROJ3 PROJ3.cpp
loli545799@ubuntu:~/Documents$ sudo ./PROJ3
[sudo] loli545799 的密码：
当前OpenSSL版本： 1.1.1f

警告：发现已知漏洞！
该版本存在漏洞238个
```

从结果我们可以得知我们的 openssl 版本号为 1.1.1f，

在过往版本中共存在 238 个漏洞。

五、结论

在本次实践中，我学习了 Python 爬虫的运作方式，c++相关程序的设计，OpenSSL 的漏洞库以及漏洞类别、检测原理。关于 OpenSSL 及其漏洞检测还存在更多需要学习和完善的地方，我将在今后补全自己的知识面。

参考文献

[1] openssl 漏洞分析

OpenSSL 安全漏洞类型以及一个重大漏洞例子分析_OpenSSL_SSL_安全漏洞_网络安全_课课家 (kokojia.com)

[2] openssl 漏洞类型

CVE security vulnerability database. Security vulnerabilities, exploits, references and more (cvedetails.com)

[3] openssl 漏洞检测方法

OpenSSL 漏洞简述与网络检测方法-CSDN 博客