Source Code Analysis

Software Security – DV2546

Sai Pratheek Vasireddy
9412145493
sava16@student.bth.se

Siddhartha Srinadhuni
9508286854
sisr16@student.bth.se

Introduction:

A set of instructions performed by a programmer to execute the specific tasks is termed as source code of a program. Source code plays a significant role in understanding the software artifacts[1]. A generic procedure of analysis is performed so as to audit the source code which in turn distinguishes the existing bugs and therefore confining and impeding the vulnerabilities. Further, source code is typically accessible and readable to the users.

For this particular assignment, we are assigned with Post Office Protocol 3 server source code and are to analyse it in order to find vulnerabilities. The task given is to exploit the found vulnerabilities and exploit them. We have also devised measures to outcome the vulnerabilities, descriptions of bugs and reflexion in the sections below.

Task 1: Vulnerability repository [2]

The Vulnerability chosen for this task as per the given instructions is CVE-2016-3858. National Vulnerability database(NVD) has been the reference for this vulnerability. It is the United States government repository of standards represented using the Security Content Automation Protocol (SCAP).

Description: Buffer over-flow in drivers/soc/qcom/subsystem_restart.c in the Qualcomm subsystem driver in android devices such as nexus 5X and 6P devices allows attackers to gain privileges via a crafted application that provides a long string

It is described as buffer errors as per CWE (common weakness enumeration) where a software performs operations on memory buffer, but it can modify the data to a memory location which is outside the boundaries of the buffer.

Countermeasures :

- Abstaining the use of functions such as strcpy(), strcat(), sprint(), vsprintf(), getwd() , gets(), realpath(), fscanf(), scanf(), sscanf() and replacing them with functions such as fgets, strncat, strncpy,snprintf,vsnprintf, decrease the chance of exploiting the vulnerabilities caused in the system[3]

- The supervision of the boundaries in the functions can be done by using the above functions
- Fixing possible overflow while copying firmware name in drivers/soc/qcom/subsystem_restart.c directory

Risks :

- Integrity
- Confidentiality
- Availability

Task 2 : Read Bob's mail

The aim of this task is to read bob's mail while having the access of alice account. This can be done by exploring vulnerabilities in the source code manually. The desired vulnerability leads to the exploitation of bob's account and the mail can be seen.

After analysing the source code pop3authenticate manually it was identified that in the function pop3authenticate as mentioned in figure 1.a , 1024 bytes was allocated to buff in a hard coded way such as char buff[1024] which is vulnerable to the program. The size of the buff can be changed in the later state of the program leading to buffer over-flow.

```
33    char * pop3authenticate( char * pUsername , int nMaxUserName )
34  1 {
35  1   int authorized = 0;
36  1   char username[32];
37  1   char buf[1024];
38  1   char * pThisPart = 0,
39  1        * pBuf;
```

Figure 1.a

The vulnerability is exploited by overflowing the buffer by printing more than the desired size 1024 as mentioned in figure 1.b and the result in mentioned in figure 1.c

Figure 1.b

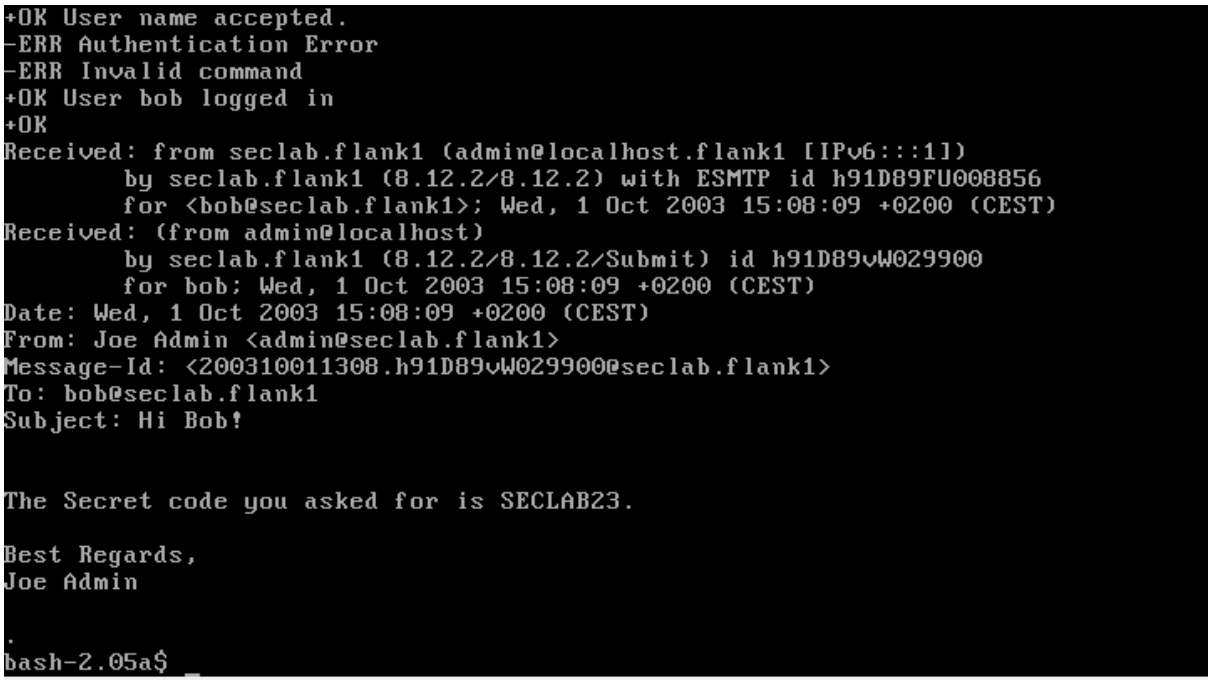

Figure 1.c

Countermeasures :

[3]The declaration of buff can be changed from char buff[1024] to #define BUFF_SIZE 1024; char buff[BUFF_SIZE] where buff is set constant throughout the program and cannot be changed eventually.

Task 3 : Prevent Bob from reading the mail

The aim of this task is to prevent bob from reading the mail which was retrieved in task 2. This can be done by source code examination manually for vulnerabilities.

After analysation of the source code manually from pop3transacion code from figure 2.a , it was identified that there were specific conditions mentioned for a particular variable msgno such as (msgno=0, msgno>0 etc) but there wasn't clear interpretation for the condition (msgno<0). This vulnerability can be exploited and we can prevent bob from reading his mail

```
int pop3transaction( void )
{
  char buf[1024];
  char * pThisPart, *pBuf;
```

Figure 2.a

From figure 2.b from task 2 in the source code "retr 1" can be changed to retr -1 by exploitation of the condition (msgno<0). This condition prevents the user from viewing his mails as shown in figure 2.c

```
#!/usr/local/bin/bash
echo "user bob"
sleep 1
echo "pass "
for ((i=0;i<1054;i++))
do
printf "z"
done
sleep 1
echo "list"
sleep 1
echo "retr -1"
```

```
-----Mg: bash1.sh                          (fundamental)----------------------
(Read 11 lines)
```

Figure 2.b

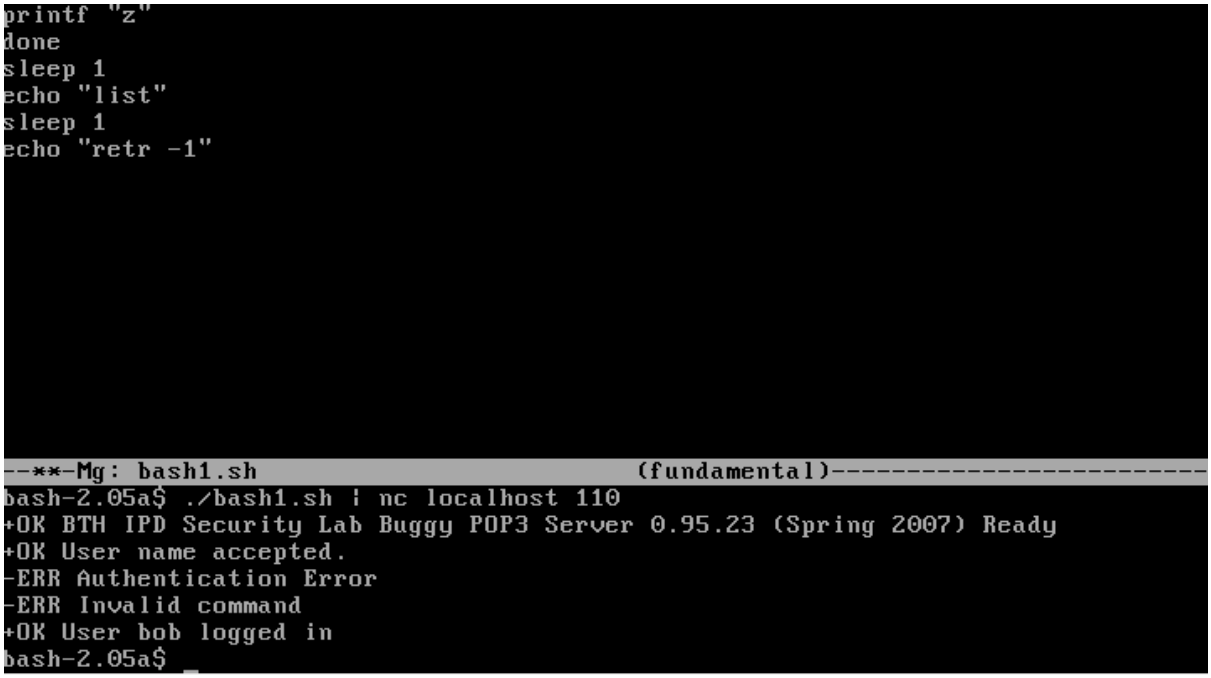

Figure 2.c

Another vulnerability identified was atoi function, depending upon the value assigned to pthispart it changes its behaviour and causes overflow , the source code is shown in figure 2.d

```
msgno = atoi( pThisPart );
```

Figure 2.d

Countermeasures :

- A specific proper condition should be interpreted for (msgno<0)
- atoi() functions can be replaced by strol() functions as assignment of values or conversion can be done precisely.

Task 4 : Prevent valid user logins

The aim of this task is to prevent any user from logging into his account.

From [4] and [5] we have inferred that spwd.db stores passwords as it is the secure password database file. Sensitive information like password hashes are stored in this file. By overflowing this file, secured hashes can be deleted which prevents any user from logging into the system. This database is located in the etc folder. ../../../etc/spwd.db path is set in the code as mentioned in figure 3.a and is being overflowed leading to the exploitation of the vulnerability . The result is shown in the figure 3.b

Figure 3.a

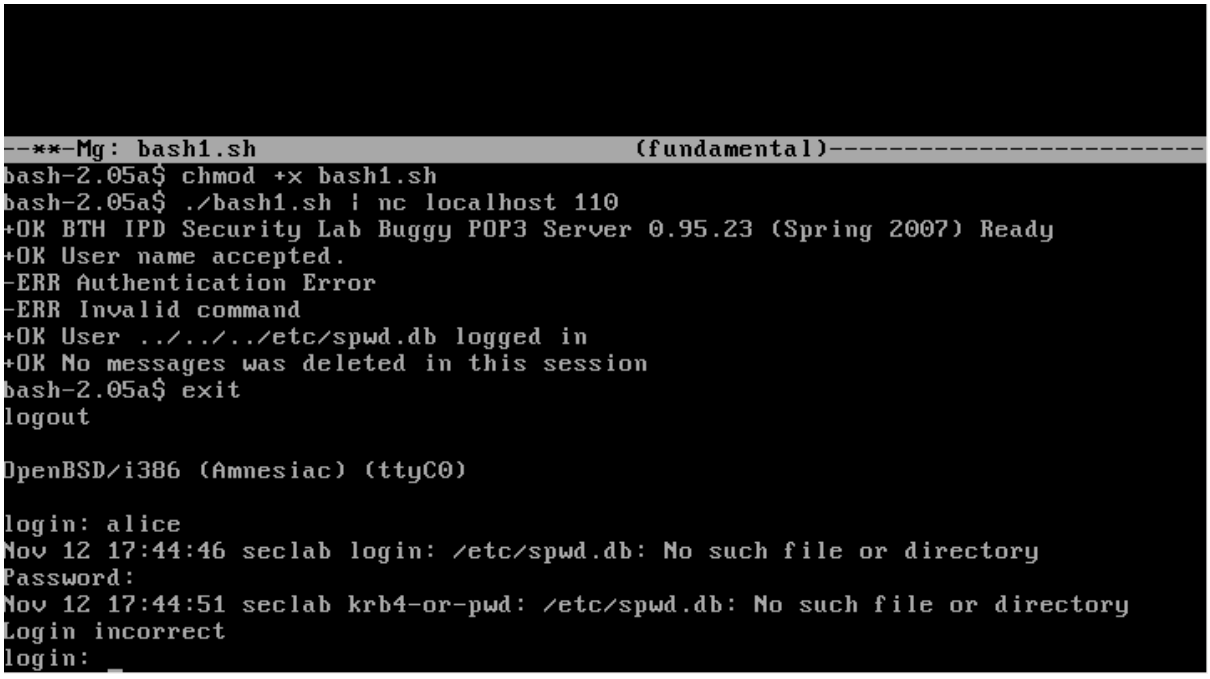

Figure 3.b

Countermeasures :

- Spwd.db database show have limited access and if any changes are to be made vipw command is used to edit it as it locks files at a time after the changes are made
- A separate copy of spwd.db patch should be stored in the master.passwd file

Task 5 : Obtaining Root Access

We could not complete the task due to insufficient knowledge.

Reflexion:

This reflexion report includes the tasks that we have broken down, amount of time invested to complete the tasks, complexity of the task and the learnings out of the tasks performed. Since we are naïve in terms of hands on experience with UNIX and taking the initial steps into security, we have started with learning the UNIX commands. The difficulty level that is portrayed below is purely based on our understandings towards the tasks. The table below illustrates the further reflexions.

S.no	Tasks	Time Invested	Learnings	Difficulty Level
1.	Source Code Analysis	2 days	Identifying vulnerabilities	Medium
2.	UNIX commands	1 day	Understanding the implementation of the commands	Medium
3.	Vulnerability Repository	5 hours	Devising countermeasures for buffer overflow	Easy
4.	Reading Bob's mail	5 hours	Implementing the buffer overflow and the learnings involved from the previous task.	Medium
5.	Preventing Bob from reading the mail	1 day	Analysing the source code and learning the implementation of new commands	Medium
6.	Preventing the valid user's login	2 days	Learnings involved locating different databases and passwords	Hard
7	Obtaining the root access	2 days	Gaining unauthorised access and effects of it. But our	Hard

			understanding wasn't sufficient to complete the task	
8	Report Writing	1 day	Reflecting our learnings and understanding with respect to the tasks given.	Easy.

References :

[1] G. Cormode, S. Muthukrishnan, and J. Yan, "Studying the Source Code of Scientific Research," *SIGKDD Explor Newsl*, vol. 14, no. 2, pp. 59–62, Apr. 2013.

[2] https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-3858#VulnChangeHistoryDiv

[3]https://developer.apple.com/library/content/documentation/Security/Conceptual/SecureCodingGuide/Articles/BufferOverflows.html

[4] https://www.freebsd.org/cgi/man.cgi?query=pwd_mkdb

[5] http://www.bsdnewsletter.com/bsda-book/Protect_authentication_data.html