Penetration Test - Exercise 07

Esteban Calvo

2023-10-02

Contents

1 Technical Report

1.1 Finding: *Netcat Shell Access*

Severity Rating

Critical Risk:
CVSS Base Severity Rating: 9.6 AV:N AC:L PR:N UI:R S:C C:H I:H A:H

Vulnerability Description

The program running on port 1337 written by Brian Oppenheimer is intended to allow system admins see the server status without having to login and thus essentially run as regular users. An error in the code allows for a buffer overflow of the name to leak into the list of possible commands to run and therefore allows the user to run a shell with elevated commands if they overflow the correct command.

Confirmation method

The following script can be used to gain shell access to the server

```
netcat www.artstailor.com 1337
1234567890123456hs
brian
sh
```

Mitigation or Resolution Strategy

To address this problem, there are two possible solutions. The easiest solution would be to disable this service or make sure it can only be accessed within internal networks. There is no real reason other than ease of use for this port to always be active and listening so I recommend the termination of this program. If there is adequate reason to keep this service running, the toool.c code must be changed to make sure that overflowed user input will not leak into the list of commands.

2 Attack Narrative

2.1 Port Discovery

The post made on 42chan by Brian revealed that there was some port on artstailor.com that could be exploited. Running the command

```
nmap -p- www.artstailor.com
```

revealed that port 1337 was open. Using the command

```
netcat www.artstailor.com 1337
```

Returned the response I wanted, the next part from here was to look for possible vulernabilites with this program.

2.2 Exploitation

The next part of this process was seeing if there was a way to exploit the program responding to my nc request. In the program, there were only three allowed commands and none of them helped in any way to gain access to the machine. My first idea was to attempt to overflow the buffer reading in the commands but I found that I was not able to truly understand at what point the buffer was overflowing and why so I shifted focus. The next part to try was the login response that asked for my name. It states that there is a 15 character limit and and I saw by typing in random numbers the program was in fact reading in 16 characters and then after that, the rest of the characters would overflow into the list of acceptable commands. Another observation I saw was that the ps auxww command was no longer able to be executed which meant that it was in fact changing the list of commands I was allowed to type. The last important observation I noticed was that after 16 characters, the characters after that were getting reversed. So, putting this all together, I came up with the idea to insert the following name when prompted

```
Enter Name of admin (max 15 characters)
1234567890123456hs
Enter Name of admin (max 15 characters)
brian
Enter Command
netstat -nat
ip a
sh
```

From here, typing the sh command gave me shell access and I was able to navigate all directories as user brian confirmed by the following

```
id
uid=1000(brian) gid=1000(brian) groups=1000(brian), 100(users)
```

2.3 Analyzing Code

The code for the netcat server response was found inside the home folder for brian "/home/brian/toool.c". Analyzing the code, I was able to see some important features such as why the characters at the end were reversed. The

most important thing to note though was the source of the overflow which was caused by fgets reading in a buffer of size 1024 while the expected size of admin name is 15 characters plus a newline character. Thus, to fix the issue, we need to rewrite this line as follows

```
fgets(admin, sizeof(admin), stdin);
```

Several other lines of code are questionable such as putting all the commands in one array, but I believe the reading of the name was the main issue that caused the overflow.

2.4 MITRE ATT&CK Framework TTPs

TA0001: Initial Access
- **T1190:** Exploit Public Facing Application

TA0043: Reconnaissance
- **T1593:** Search Open Websites/Domains
  - **.001:** Social Media

TA0002: Execution
- **T1059:** Command and Scripting Interpreter
  - **.004:** Unix Shell

2.5 Key

With access to the home folder for brian, I was able to ls -la and see a file called ".secret". Inside this file was contained the following key

```
KEY009-\x80\xa9\x86\xbc\xa6\xb7\x80\xa6\x92\xa1\x9f\x95
\xc7\x9d\xc0\x9e\xbc\x9e\xb2\x85\xb7\x86\xcc\xcc
```

The key is encrypted, but no successful attempt decrypted the key.