**Report**

**A Practical Ethical hacking:**

**Buffer-overflow attack**

**Group No. 8**

Specification : Ubuntu 18.04 32 bit

|  |  |  |
| --- | --- | --- |
| Member Name | Roll No | Contribution |
| Adrija Chakrabotry | 2020201063 | 25% |
| Mahak Modani | 2020201068 | 25% |
| Nayanika | 2020201056 | 25% |
| Rani Tresa | 2020202019 | 25% |

**Q1 What is Stack guard? What is ASLR protection?**

It is a simple compiler technique that virtually eliminates buffer overflow vulnerabilities with only modest performance penalties. Privileged programs that are recompiled with the StackGuard compiler extension no longer yield control to the attacker, but rather enter a fail-safe state. These programs require no source code changes at all, and are binary-compatible with existing operating systems and libraries.. Privileged programs that are recompiled with the StackGuard compiler extension no longer yield control to the attacker, but rather enter a fail-safe state.

Address Space Layout Randomization (ASLR) is a security technique used in operating systems, first implemented in 2001. Address space layout randomization (ASLR) is a memory-protection process for operating systems (OSes) that guards against [buffer-overflow](https://searchsecurity.techtarget.com/definition/buffer-overflow) attacks by randomizing the location where system [executables](https://searchsecurity.techtarget.com/definition/executable) are loaded into [memory](https://searchstorage.techtarget.com/definition/memory-card).The success of many cyberattacks, particularly [zero-day exploits](https://searchsecurity.techtarget.com/definition/zero-day-vulnerability), relies on the hacker's ability to know or guess the position of processes and functions in memory. ASLR is able to put [address space](https://searchstorage.techtarget.com/definition/address-space) targets in unpredictable locations. If an attacker attempts to exploit an incorrect address space location, the target application will crash, stopping the attack and alerting the system.

**Q2.Perform a stack overflow attack on the stack.c and launch shell as root under when Stack is executable stack and ASLR is turned off.**

Ubuntu and several other Linux-based systems use address space layout randomization to randomize the starting address of the heap and the stack. This makes guessing exact addresses in these regions difficult; guessing addresses is one of the critical steps of buffer-overflow attacks. we disable this feature using the command.

**StackGuard Protection** : GCC implements a security mechanism called StackGuard to prevent buffer overflows. In the presence of this protection, buffer overflow attacks will not work. We can disable this protection during compilation using the -fno-stack-protector option

Before starting the attack, let's get familiar with *shellcode*. Shellcode named because it is code injected into a program typically used to launch a shell. It has to be loaded into memory so that we can force the vulnerable program to jump to it.The shellcode invokes the **execve**() system call to execute /bin/sh.

stack.c program has a buffer overflow vulnerability. It first reads an input from a file called badfile, and then passes this input to another buffer in the function bof().

To compile the vulnerable program, do not forget to turn off StackGuard and the non-executable stack protections using the -fno-stack-protector and -z execstack options. After compilation, we need to make the program a root-owned Set-UID program. We can achieve this by first changing the ownership of the program to root , and then changing the permissions to 4755 to enable the Set-UID bit.

Task 2 : Exploit the vulnerability

**Q3. Perform a stack overflow attack on the stack.c and launch shell as root and perform seteuid() under when Stack is executable stack and ASLR is turned off.**

Many commands will behave differently if they are executed as a Set-UID root process, instead of just as a root process, because they recognize that the real user id is not root. To solve this problem, you can run the program to turn the real user ID to root. This way, you'll have a real root process, which is much more powerful.

**Q4. Perform a stack overflow attack on the stack.c and kill all processes when Stack is executable stack and ASLR is turned off. It is a kind of Denial of Service attack.**

A Denial-of-Service (DoS) attack is an attack meant to shut down a machine or network, making it inaccessible to its intended users.We can kill all the processes using shellcode.Do not forget to turn off StackGuard and the non-executable stack protections using the -fno-stack-protector and -z execstack options.

**Q5. Perform a stack overflow attack on the stack.c and reboot the system when Stack is executable stack and ASLR is turned off**

**Q6. Now turn on ASLR and perform all the tasks from 2 to 5.**

**Q.7 Turn on a non-executable stack . Perform any ret2libc attack.**

Ubuntu used to allow executable stacks (i.e. the processor can fetch instructions from the stack), but this has now changed: the binary images of programs (and shared libraries) must declare whether they require executable stacks or not, i.e., they need to mark a field in the ELF program header. The kernel and dynamic linker use this mark to decide whether to make the stack of this running program executable or non-executable. This marking is done automatically by recent versions of gcc, and by default, stacks are set to be non-executable.

A "return-to-libc" attack is a computer security attack usually starting with a buffer overflow in which a subroutine return address on a call stack is replaced by an address of a subroutine that is already present in the process executable memory, bypassing the no-execute bit feature (if present) and ridding the attacker of the need to inject their own code.

**Q8.Exploit the heap and try to execute the executeShell function to launch a shell.**

To practice exploiting a very simple heap overflow vulnerability. This one is easy to exploit because there's a pointer in the heap that is used for a function call. That makes a heap overflow as simple as a stack overflow targeting EIP.