Fabian Hupfeld (35523329) und Konstantin Stitz (35532242)

Übungsaufgaben 2

Aufgabe 1.)

Siehe extra-PDF

Aufgabe 2.)

a.)

Bei dem Programm kann der Buffer-Overflow-Exploit ausgenutzt werden, dabei wird mithilfe der strcpy-Funktion ein Buffer-Overflow erzeugt, mit welchem die Rücksprungadresse der Funktion auf dem Stack geändert wird.

Wenn die Funktion check aufgerufen wird, wird auf dem Stack die Rücksprungadresse gespeichert, wo das Programm fortfahren soll, wenn die Funktion beendet wurde. Man kann nun die Funktion mit einem pointer (*password) aufrufen, welcher zu einem Buffer-Overflow führt, da in den password_buffer maximal 16 Zeichen passen, aber nirgends überprüft wird, dass der Parameter diese nicht überschreitet. Durch den Buffer-Overflow wird dann auf Speicherbereiche außerhalb des Arrays zugegriffen, wodurch man durch geschicktes wählen des Parameters die Rücksprungadresse auf dem Stack so ändern kann, dass das Programm nach dem Funktionsaufruf nicht korrekt zurückspringt, sondern in die if-Schleife springt, die eigentlich nur ausführt werden sollte, wenn das Passwort korrekt ist.

b.)

Ein Segmentation Fault/Access Violation kann bei einem Buffer-Overflow entstehen, wenn dadurch auf Speicherplatz zugegriffen wird, auf das das Programm keinen Zugriff hat. Beispielsweise wenn der Speicherbereich Read-Only ist, oder anderweitig geschützt ist. In dem vorliegenden Programm tritt der Fehler auf, wenn das übergeben Passwort sehr lang ist, und dadurch versucht wird auf einen Speicherbereich der nicht dem Programm zur Verfügung steht zuzugreifen.

c.)

Man muss verhindern, dass es zu einem Buffer-Overflow kommen kann, also vorher abfragen, wie lang der String ist, der in im password_buffer gespeichert werden soll. Nur wenn dieser dann auch in das Array passt, soll strcpy aufgerufen werden, damit keine anderen Speicherbereich verändert werden können. Dies könnte man zum Beispiel mit einer einfachen if-Abfrage umsetzen, welche mit .length() überprüft ob der String <= 16 Zeichen ist. Man könnte auch die strncpy-Methode verwenden, bei welcher als 3. Parameter noch die maximale Anzahl der zu kopierenden Bytes übergeben werden kann, jedoch kann dies zu Problemen führen, wenn der übergebene String zu groß ist, da dann die Nullterminierung fehlt.

Aufgabe 3.)

a.)

- Standalone Systemvirtualisierung, dabei setzt die virtuelle Maschine direkt auf der Hardware auf, was zum Beispiel bei der Virtualisierung in großen Serveranlagen verwendet wird. Ein Beispiel wäre VMware ESXi.

- Systemvirtualisierung mit einem Host-Betriebssystem, dabei setzt die Virtualisierungssoftware nicht direkt auf der Hardware auf, sondern läuft als Programm in dem Host-Betriebssystem. Ein Beispiel wäre Virtualbox.

- Prozessvirtualisierung über eine Anwendungs-VM, dabei werden nicht ganze Betriebssysteme virtualisiert, sondern einzelne Programme, wie es zB bei der JVM für Java der Fall ist.

- Betriebssystemvirtualisierung, wobei es in dem Host-Betriebssystem mehrere Gast-Systeme (Container) gibt, das Gast-System muss jedoch mit dem Host-System kompatibel sein, da sie denselben Kernel verwenden. Ein Beispiel wäre LXC, wobei eine eigene Umgebung mit eigenen Prozessen erzeugt wird, jedoch der Kernel des Linux-Host-BS verwendet wird.

b.)

Vorteile	**Nachteile**
Portabilität -> Software kann auf allen Rechnern laufen für die es die VM gibt	langsamer (schlechtere Performance), da zusätzliche Schicht zwischen Soft- und Hardware
Software kann auf gleichem Rechner mit unterschiedlichen Betriebssystemen getestet werden	höherer Speicher-/ Ressourcenverbrauch
Sicheres Testen neuer Software, da Fehler nicht zu Computerabsturz etc. führen, allgemein höhere Sicherheit -> „Sandbox-Modus“	Keine individuelle Anpassung/ Optimierung auf die einzelnen Betriebssysteme
automatische Speicherverwaltung	