Buffer Overflow - Angriffe und Gegenmaßnahmen

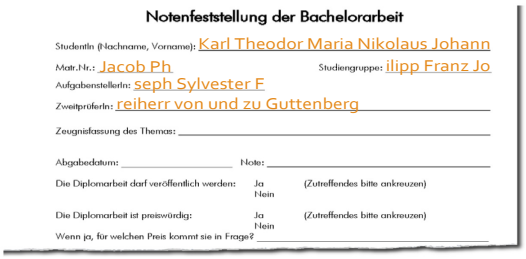
Dr. Alexander Schinner

Abstract

Pufferüberläufe gehören zu den häufigsten Sicherheitslücken in aktuellen IT-Systemen. Es wird versucht zu beschreiben, wie sie entstehen, wie man die direkten Folgen unterdrücken kann und mit welchen Maßnahmen man das Gesamtsystem schützen kann.

1 Buffer Overflow

Variablen eines Computerprogramms sind die Behälter für Rechnungsgrößen („Werte“), haben typischerweise eine feste Adresse und Länge. Dabei liegen sie im Speicher sehr nahe beisammen. Versucht man nun, mehr Daten in eine Variable zu schreiben, als sie Platz bietet (Buffer Overflow), kann es also passieren, dass die Nachbarn überschrieben werden. Bildlich kann man sich das gut anhand eines Formulars vorstellen, bei dem dann am Ende eines Feldes im folgenden Feld weitergeschrieben wird. (vgl. Abb. 1). Damit kann ein Angreifer u.U. Werte verändern, die für ihn nicht zugänglich sein sollten.



Notenfeststellung der Bachelorarbeit

StudentIn (Nachname, Vorname): Karl Theodor Maria Nikolaus Johann

Matr.Nr.: Jacob Ph Studiengruppe: ilipp Franz Jo

Aufgabenstellerin: seph Sylvester F

ZweitprüferIn: reiherr von und zu Guttenberg

Zeugnisfassung des Themas: _____

Abgabedatum: _____ Note: _____

Die Diplomarbeit darf veröffentlich werden: Ja Nein (Zutreffendes bitte ankreuzen)

Die Diplomarbeit ist preiswürdig: Ja Nein (Zutreffendes bitte ankreuzen)

Wenn ja, für welchen Preis kommt sie in Frage? _____

Abb. 1: Buffer Overflow in einem Formular

2 Funktionsaufrufe in C und C++

Auch wenn die genauen Aufrufkonventionen von Unterprogrammen in C und C++ von verschiedenen Faktoren (CPU, Compiler, Betriebssystem) abhängen, haben sie eine wichtige Eigenschaft gemeinsam: Die Rücksprungadresse und die lokalen Variablen liegen nebeneinander auf dem Stack. Das bedeutet, dass ein Angreifer durch einen Buffer Overflow die lokalen Variablen ändern kann und durch Überschreiben der Rücksprungadresse Einfluss auf die Befehlsausführung (Shell Code) bekommt.

Shell Code ist dabei eine Folge von Befehlen, die in ein Programm eingeschleust werden und typischerweise eine lokale Shell starten oder an einen Port binden; daher auch der Name. Selbstverständlich sind auch andere bösartige Aktionen wie das Hinzufügen neuer Accounts zum System (root/administrator) oder das Ändern von Dateirechten (z.B. /etc/shadow) möglich.

3 Schützen von Buffer Overflows

Geht man davon aus, dass man Buffer Overflows nicht wirklich verhindern kann, so kann man versuchen, den Aufwand für den Angreifer zu erhöhen. Alle drei folgenden Maßnahmen erschweren die Nutzung von Shell Code, schützen aber vor der Änderung von lokalen Variablen nur bedingt.

3.1 Stack Canary

Indem ein Zufallswert (Stack Canary) zwischen den lokalen Variablen und der Rücksprungadresse gespeichert wird, kann man am Ende der Routine einen Buffer Overflow an einer Änderung dieses Wertes erkennen.

3.2 Memory Address Randomization

Durch die zufällige Platzierung von Stack und Heap im Speicher wird es dem Angreifer sehr schwer gemacht, die Sprungadressen für den Shellcode zu erraten.

3.3 Hardwareansätze

Mit Unterstützung des Betriebssystem und der MMU (Memory Managing Unit) können bestimmte Speicherbereiche so markiert werden, dass hier kein Programmcode (also auch Shell Code) ausgeführt werden kann.

4 Schutz vor Buffer Overflows

Im Sinne der Maxime „Defense in Depth“ müssen aber weitere Maßnahmen getroffen werden um ein produktives System zu schützen.

4.1 Schutz bei der Entwicklung

Um das Entstehen von Buffer Overflows bei der Entwicklung zu verhindern müssen Entwicklern Richtlinien vorgegeben werden und sie brauchen ausreichend Zeit für Tests.

4.2 Schutz auf Betriebssystemebene

Man muss immer davon ausgehen, dass ein Dienst durch einen Angreifer übernommen werden kann. Deswegen sollen Maßnahmen wie Sandboxing oder User mit reduzierten Rechten den Angreifer einschränken.

4.3 Schutz auf Netzwerkebene

Heutzutage spielen sich die typischen Angriffe über das Netzwerk ab. Deswegen müssen auch hier Schutzmaßnahmen wie Firewall, IDS und WAF zu Einsatz gebracht werden.

4.4 Schutz auf organisatorischer Ebene

Durch die Überwachung von Diensten, ein geregeltes Patchmanagement und festgelegte Verantwortlichkeiten kann der Geschäftsbetrieb zusätzlich geschützt werden.

4.5 Schutz auf Notfallebene

Eine gute Planung geht auch vom schlimmsten Fall aus und versucht festzulegen, wie der Geschäftsbetrieb wieder fortgesetzt bzw. wieder aufgenommen werden kann.

Literaturverzeichnis

[1] Aleph One, “Smashing the stack for fun and profit”, Phrack 49