Buffer Overflows

Praktische Analyse von Schwachstellen

Jakob Stühn, John Meyerhoff, Sam Taheri

H-BRS

January 10, 2022

Inhalt

- Geschichte
- Grundlegende Theorie
  - Speicheraufbau
  - Stack-/Heap-Overflow
- Shellcode
- Praktische Analyse
  - Programmierfehler
  - Demonstration
- Gegenmaßnahmen
- Fazit

Geschichte: Bekannte Buffer Overflows

- The Morris Worm (November 1988)
- SQL-Slammer (Januar 2003)
- HP-Drucker Firmware (2018)
- WhatsApp MP4 (2019)

Im weitesten Sinne beschreibt ein Buffer Overflow eine Schwachstelle in einem Computerprogramm, bei der ein Angreifer einen Speicherbereich fester Größe überschreibt und diesen so zum "Überlaufen" bringt. Durch Ausspähen und Analysieren der Software kann dieses Überschreiben so gezielt geschehen, dass der Fluss des Programms verändert und zuvor injizierter Schadcode ausgeführt wird.

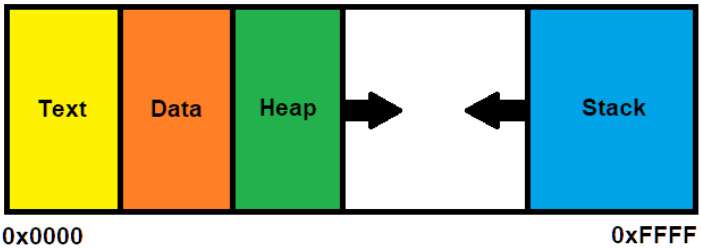

Figure: Prozess im Speicher

- Wert einer Variable verändern
- Function Pointer manipulieren
- Return Pointer überschreiben

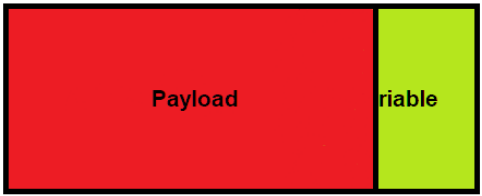

Figure: Buffer im Stack während eines Overflows

Gegenmaßnahmen

Struktur

- Stack-Schutz mit “Canary” (Zufallszahl)
- Safe Pointer Instrumentalisierung
- C Range Error Detector und Out Of Bounds Object
- Hardware-basierte Lösungen
- Statische Code-Analyse
- Betriebssystembasierte Ansätze
- Manuelles Buffer-Overflow Blocken (Input-Bereinigung)

Gegenmaßnahmen

Code-Beispiel

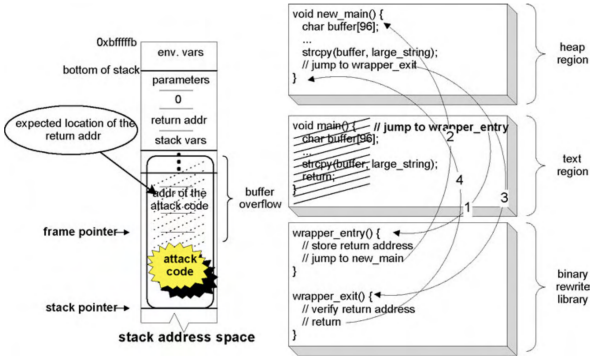

Figure 9: Libverify funtion call and stack layout

Gegenmaßnahmen

Testen

- Fuzzy Tests
- Spezifische Payloads