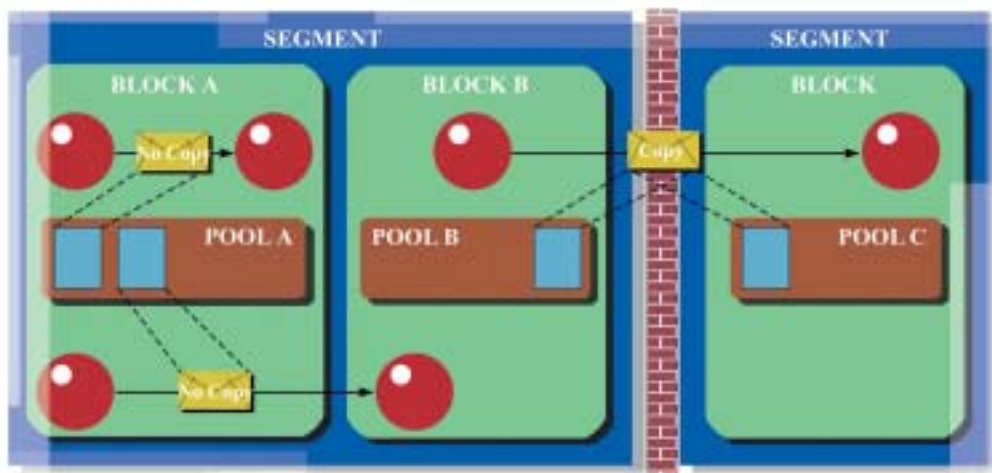


Hardware-Speicherschutz verhindert Zugriffsverletzungen

Wilde Zeiger finden und vermeiden

Sie sind überall und kommen ohne Vorwarnung, sind hartnäckig und können Softwareentwickler zur Verzweiflung bringen. Die Rede ist von sogenannten ‚wilden Zeigern‘, wie unbeabsichtigte Referenzen auf fremde Speicheradressen im Fachjargon genannt werden. Wie lassen sich diese effizient bändigen? Hardware-MMUs und ein intelligentes Betriebssystem können helfen. JÖRG DORN



Interprozesskommunikation im geschützten System



Dipl.-Ing. (FH) JÖRG DORN
ist für den Echtzeitbetriebs-
systemhersteller OSE
Systems GmbH als
Softwareingenieur tätig.

Gerade in der verbreiteten, zeigerorientierten Programmiersprache C/C++ gibt es vielfältige Ursachen, die dazu führen, dass ein Programm auf Speicher zugreift, der dafür nicht vorgesehen, also allokiert ist. Das reine Auslesen solcher Speicherbereiche richtet wenig Schaden an und der Fehler lässt sich normaler-

weise leicht finden und beheben. Allerdings kann es passieren, dass zunächst kein Fehlverhalten auftritt, da der Speicher noch nicht wiedervergeben wurde und somit die erwarteten Werte enthält. Bei höherer Auslastung des Systems kann sich dies dann ändern. Weitaus komplizierter ist es, wenn auf den von anderen Programmen allokierten Speicher geschrieben

wird. Die Folge kann eine Veränderung des betroffenen Programmcodes sein. In den meisten Fällen wird dann dieses Programm abstürzen, da die Daten nicht mehr als ausführbares Programm interpretierbar sind.

Die häufigste und harmloseste Form wilde Zeiger zu erzeugen, ist das Schreiben über das Ende eines angeforderten Speicherbereichs in einem Puffer oder einem Array. Dieser Fehler kann vermieden werden, wenn das Betriebssystem Ende-Kennungen einfügt und überprüft. Problematischer sind Zugriffe auf bereits freigegebenen Speicher oder falsche Berechnungen von Indizes oder Offsets. Der Grund des Übels lässt sich oft nicht einfach auffindig machen und in den Büros der Softwareentwicklung geht dann die bange Frage um: „Wer ist Schuld“?

Nur schwer auffindbar

Wilde Zeiger gehören zu den am schwersten auffindbaren Softwarefehlern. Der Grund dafür ist, dass von der Wirkung (Speicher ist mit falschen Werten überschrieben) meist nicht auf die Ursache, also die Codezeile, die den Zugriff bewirkt, geschlossen werden kann. Da alle Module prinzipiell Zugriff auf den gesamten Speicherbereich haben, ist eine Einschränkung der Fehlerquelle auf Teile der Software nicht möglich. Zudem muss in dem verursachenden Modul nicht zwingend ein Fehlverhalten festzustellen sein. Zu allem Überfluss treten derartige Probleme meist nur unter nicht reproduzierbaren Umständen auf; häufig auch erst gegen Ende des Entwicklungszyklus, wenn alle Module integriert sind und unter realen Bedingungen getestet werden. Und manchmal tritt der Fehler erst im praktischen Einsatz, also beim Kunden, auf. Dann ist guter Rat buchstäblich teuer: Die einzige analytische Methode, wilden Zeigern auf die Spur zu kommen, ist der Einsatz eines Logik-Analysators, der auf die korrupte Speicheradresse triggert. Anschließend kann der Programmierer aus den vorangegangenen Adressen auf die Codezeile

schließen, die zuletzt den Speicher beschrieben hat. Voraussetzung ist, dass der Fehler überhaupt nachgestellt werden kann. Auch der Anschluss des Analysators an Adress- und Datenbus ist meist kein leichtes Unterfangen.

Murphys Gesetz entgehen

Ziel muss es also sein, die wilden Zeiger schon im Vorfeld zu identifizieren, um den fehlerhaften Zugriff zu verhindern bevor Schaden entsteht. Dafür lassen sich Funktionen vorstellen, die bei der Allokation und bei jedem Speicherzugriff aufgerufen werden und die zulässigen Grenzen überprüfen. Nachteilig ist vor allem der Performance-Verlust, der umso schmerzlicher ist, da der Programmierer den effizienten Zugriff über Zeiger auf die Daten schätzt. In einem etwas verfeinerten Ansatz könnte er den langsamen aber sicheren Zugriff in der Testversion einsetzen, um dann die endgültige Version ohne ‚Bremse‘ zu erzeugen. Leider sind wilde Zeiger aber nicht so leicht einzufangen. Leidvolle Erfahrungen zeigen, dass ganz nach Murphys Gesetz „If anything can go wrong – it will“ der Fehler häufig erst in der Auslieferungsversion auftritt. Die Laufzeitveränderungen bringen zusätzliche, nachteilige Effekte.

Hilfe von der MMU

Was also tun? Wie immer, wenn Softwarelösungen an der Performance scheitern, macht es Sinn, auf die Hardware zurückzugreifen. So sind in fast allen modernen Mikrocontrollern leistungsfähige Memory Management Units (MMUs) integriert, die eine Überwachung der Speicherzugriffe übernehmen können. Hierbei wird für jede Adresse, auf die zugegriffen werden soll, ein Eintrag in einer Attributtabelle überprüft. Da dies innerhalb der CPU-Hardware geschieht ist die Verzögerung minimal. Zudem werden die Einträge im Cache abgelegt, was wiederum Laufzeitvorteile bringt.

Neben dem Schutzmechanismus ist die MMU auch für die Cache-Eigenschaften und gegebenenfalls für das Übersetzen von effektiven Adressen in physikalische Adressen verantwortlich. Der Speicherschutz, der mit MMU erreicht werden kann, lässt sich in zwei Kategorien unterteilen: Zum einen können für einzelne Speicherbereiche Zugriffsattribute vergeben werden, zum anderen lässt sich durch Austausch der Attributtabelle ein Schutz zwischen einzelnen Speicherbereichen schaffen.

Die Attribute sind statisch und werden zum Systemstart konfiguriert. Trotzdem ist auch hier schon die Implementierung eines effizienten Speicherschutzes möglich. Zunächst wird zwischen Instruktions- und Datenzugriff unterschieden. Dabei kann ein Attribut vergeben werden, das für diesen Bereich keinen Instruktionszugriff zulässt. Weiterhin sind die Zugriffsrechte daran gebunden, ob der zugreifende Prozess im Supervisor-Modus oder im Anwendermodus ausgeführt wird. Diese Eigenschaft verwaltet normalerweise das Betriebssystem. Mit den Speicherattributen können die Zugriffe getrennt für den Supervisor-Modus und User-Modus zum Lesen oder Schreiben gesperrt werden. Somit ergeben sich zehn unterschiedliche Zugriffsrechte, die für einen Bereich konfiguriert werden können (Tab. 1).

Der Programmcode wird ausschließlich für den lesenden Zugriff freigegeben. Nachzunutzender Code, wie er zum Beispiel in den Standardbibliotheken zu finden ist, muss von allen User-Prozessen ausführbar sein. Andere Bereiche, wie das Betriebssystem, können vor dem Zugriff nichtberechtigter Anwendungen geschützt werden. Die Daten müssen in Abhängigkeit von ihrem Geltungsbereich für alle User-Prozesse oder nur für einige Supervisor-Applikationen zugänglich sein. Um ein solches System zu konfigurieren, muss die Informationsdatei des Linkers, in dem die Sektionen für Programmcode (.text), initialisierte Daten (.data, .sdata) sowie uninitialisierte Daten (.bss, .sbss) abgelegt sind, auf die Konfiguration der MMU abgestimmt werden. Dies führt zu dem gewünschten

Tabelle 1: Speicherschutz über Zugriffsattribute

Zugriffsattribute			U-Mode		U-Mode	SV-Mode		SV-Mode
Ausführbar	Lesen	Schreiben	Instruktion	Daten	schreiben	Instruktion	Daten	schreiben
Ja	SV	SV	-	-	-	Ja	Ja	Ja
Nein	SV	SV	-	-	-	-	Ja	Ja
Ja	SV	Nein	-	-	-	Ja	Ja	-
Nein	SV	Nein	-	-	-	-	Ja	-
Ja	SV&U	SV	Ja	Ja	-	Ja	Ja	Ja
Nein	SV&U	SV	-	Ja	-	-	Ja	Ja
Ja	SV&U	SV&U	Ja	Ja	Ja	Ja	Ja	Ja
Nein	SV&U	SV&U	-	Ja	Ja	-	Ja	Ja
Ja	SV&U	Nein	Ja	Ja	-	Ja	Ja	-
Nein	SV&U	Nein	-	Ja	-	-	Ja	-

Tabellenlegende:
 SV: Supervisor-Mode
 -: Zugriffsversuch löst Ausnahmebedingung aus
 U: User-Mode
 Ja: Zugriff erlaubt

Speicherschutz. Eine strukturierte Entwicklungsumgebung kann hier in Verbindung mit einem Betriebssystem die Aufgabe erleichtern.

Schutz vom Betriebssystem

Unabdingbar wird der Einsatz des Betriebssystems bei der zweiten, noch effektiveren Form des Speicherschutzes: Diese geschieht dynamisch und schützt auch die Applikationen im Anwendermodus untereinander. In Abhängigkeit vom gerade auszuführenden Kontext können bestimmte Bereiche gelesen oder beschrieben werden, während andere gesperrt sind. Dazu müssen die Tabellen, in denen die Rechte konfiguriert sind, beim Prozesswechsel aktualisiert werden. Die Performance wird dabei nur unwesentlich beeinflusst, wenn zwischen verschiedenen Tabellen umgeschaltet werden kann. Jedoch gilt es noch eine weitere Schwierigkeit in den Griff zu bekommen: Wie können geschützte Applikationen Informationen mit anderen Applikationen und der Peripherie austauschen, ohne ihren Schutz zu verlieren? Für die Interprozesskommunikation kann das Betriebssystem eingreifen: Während Informationen innerhalb eines Bereiches sehr effizient durch das Transferieren einer Referenz weitergegeben werden, erkennt das System automatisch, wenn der geschützte Bereich verlassen wird und kopiert die Daten in den Zielbereich um (Abb. 1). Da das Betriebssystem im Supervisor-Modus ausgeführt wird, besitzt es die

Rechte dazu. Innerhalb des neuen Bereichs lässt sich dann wieder die schnelle Methode des Weitergebens von Zeigern anwenden.

Über Zusatzinformationen in jeder Nachricht, die den Sender und Empfänger kennzeichnen, kann das Betriebssystem auch hier noch eingreifen, um die Informationen eindeutig einem Besitzer zuzuweisen, der alleine einen gültigen Zeiger erhält. Diese Funktionalität erfordert allerdings eine gewisse Disziplin des Softwareentwicklers. Denn da der HW-Schutz hier nicht mehr greifen kann, ist es möglich, alle Sicherheitsmechanismen zu umgehen.

Beim Zugriff auf die Peripherie lässt sich das sogenannte ‚Trap-Interface‘ nutzen, das auf allen gängigen Prozessoren zur Verfügung steht. Ein Softwarebefehl löst eine Ausnahmebedingung aus, die das Betriebssystem wiederum behandelt. Spezielle Übergabeparameter sorgen für den Aufruf der richtigen Funktion. So ist es möglich, aus jedem User-Prozess wichtige Funktionen der im Supervisor-Modus laufenden Gerätetreiber wie ‚read‘ und ‚write‘, sowie verschiedene Konfigurationsfunktionen auszuführen. Das Betriebssystem muss dafür sorgen, dass die entsprechenden Funktionen installiert sind, sowie eine einheitliche Schnittstelle für den Anwender zur Verfügung steht. Auch hierbei steht die Leistungsfähigkeit im Vordergrund. Gerade beim Zugriff auf Datenströme gilt es, das Kunststück zu vollbringen, einen sicheren Zugriff und eine geschützte Weiterverarbeitung zu gewährleisten ohne den Ablauf zu sehr ‚auszubremsen‘.

Applikationen dynamisch nachladen

Mit den genannten Methoden ist es möglich ein System zu entwickeln, in dem verschiedene Softwarebereiche gegeneinander geschützt, nebeneinander auf einem System ausführbar sind. Wünschenswert wäre es noch, dynamisch weitere Applikationen nachzuladen oder bestehende auszutauschen, ohne andere, wesentliche Komponenten abschalten zu müssen. Dies ist mit Hilfe intelligenter Betriebssysteme problemlos möglich, die den gesamten Speicherbedarf einer Applikation dynamisch allokalieren und bei Bedarf wieder freigeben können. Zum Beispiel lassen sich damit auf Mobiltelefone immer neue Anwendungen wie Spiele laden und ausführen ohne befürchten zu müssen, dass das Adressbuch gelöscht oder der zum Telefonieren benötigte Protokoll-Stack beeinflusst wird.

Ein Hardware-Speicherschutz in Verbindung mit einem intelligenten Betriebssystem kann zwar die Auswirkungen von fehlerhaftem Speicherzugriff begrenzen und über Fehlermechanismen den Verursacher bestimmen. Der Fehler selbst jedoch bleibt bestehen und wird in der verursachenden Applikation zu spüren sein. Es bleibt also dabei, dass ein durchdachtes Softwaredesign und realitätsnahe Testläufe nicht zu ersetzen sind. Mit den richtigen technologischen Grundlagen geht es jedoch wesentlich einfacher und schneller und die MMU bietet eine Notbremse, falls trotzdem etwas schief geht.

www.publish-industry.net
more @ click DV92451