DECOIT®

Softwaredesign für dynamische Integritätsmessungen bei Linux

M. Jahnke · T. Rix · K.-O. Detken (DECOIT GmbH)
A. Rein · M. Eckel (Huawei Technologies)

Prof. Dr. Kai-Oliver Detken
DECOIT GmbH
Fahrenheitstraße 9, D-28359 Bremen
https://www.decoit.de
detken@decoit.de

DECOIT®

Agenda

- Einleitung
  - Kooperationsprojekt
  - Verwendete Technologien
  - Problemstellung
  - Dynamische Integritätsmessung
- DRIVE Framework
  - Messung
  - Referenzwertgenerierung
  - Verifikation
- Guidelines mit Beispiel
- Fazit

Einleitung
Kooperationsprojekt

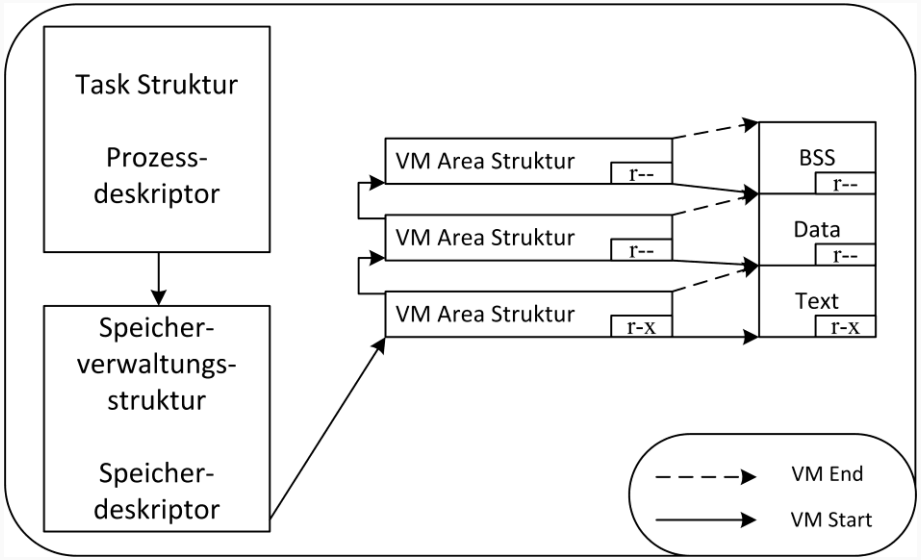
- Industrielles Forschungsprojekt zwischen der DECOIT® GmbH und Huawei Technologies: *Dynamic Runtime Integrity Verification and Evaluation (DRIVE)*
- Beide Firmen sind Mitglied der Trusted Computing Group (TCG)
- Integritätsmessungen waren im Vorfeld Bestandteil verschiedener Forschungsprojekte der DECOIT® GmbH (u.a. VOGUE, ESUKOM, SIMU)
- Dabei stand die Nutzung eines TPM-Chips im Vordergrund
- Das DRIVE-Projekt schaffte die Möglichkeit die bisherigen Forschungsansätze in die Praxis zu überführen
- Erste Ergebnisse wurden auf dem TCG Members Meeting im Juli präsentiert

Einleitung
Verwendete Technologien

- Das Trusted Platform Module (TPM) ist ein Chip der TCG, der Schlüsselmaterial bereithält, um Geräte sicher zu authentifizieren
- Das Schlüsselmaterial besteht aus
  - Endorsement Key (EK)
  - Storage Root Key (SRK)
- Der TPM kann Messungen speichern und signieren
- TPM-Funktionalität
  - Root of Trust / Hardware-Anker
  - Systemidentität
  - Integritätsmessung
  - Trusted Boot

Einleitung
Verwendete Technologien

- Speichermanagement unter Linux
  - Beim Booten reserviert der Linux-Kernel einen Teil des Arbeitsspeichers für den eigenen Bedarf: *Kernelspace Memory*
  - Der restliche Speicher steht Programmen zur Verfügung: *Userspace Memory*
  - Damit der Userspace einfacher verwaltet werden kann, nutzt der Kernel verschachtelte Datenstrukturen (siehe Abb.)
  - Virtual Memory Area (VM Area) enthält verschiedene Segmente des Prozessspeichers
  - Die VM Area ist in Pages fester Größe unterteilt



Einleitung
Verwendete Technologien

- Statische Integritätsmessung wird normalerweise durch *Integrity Measurement Architecture (IMA)* ermöglicht
- IMA ist eine Erweiterung des Linux-Kernels und ermittelt geänderte Programme
- Dafür verwaltet IMA ein Messprotokoll mit Prüfsummen:
  - Programme, Bibliotheken (ausführbare Prozesse)
  - Konfigurationen (strukturierte Daten)
- Das Messprotokoll ist mittels TPM abgesichert
  - Jede Prüfsumme wird an den TPM gesendet
  - Hashwert wird gebildet und in einem Register abgelegt
  - Änderung des Hashwertes lässt Kompromittierung erkennen

Einleitung
Problemstellung

- Angriffe werden immer intelligenter: Laufzeitangriffe nehmen zu
- Diese Angriffe zielen auf Modifikation von Programmen nach ihrer Ausführung im Speicher
- Statische Integritätsmessung überwachen ausführbare Programme direkt auf Änderungen im Programmcode
- Im Linux-Kernel wird dies durch *Integrity Measurement Architecture (IMA)* umgesetzt
- Dieser misst Userspace- und/oder Kernelspace-Programme bevor diese ausgeführt werden
- Viele Systeme werden aber nur einmal gestartet (z.B. beim Router)
- Problem:
  - Prüfung nur beim Start
  - Modifikationen nach dem Start der Software werden nicht erkannt

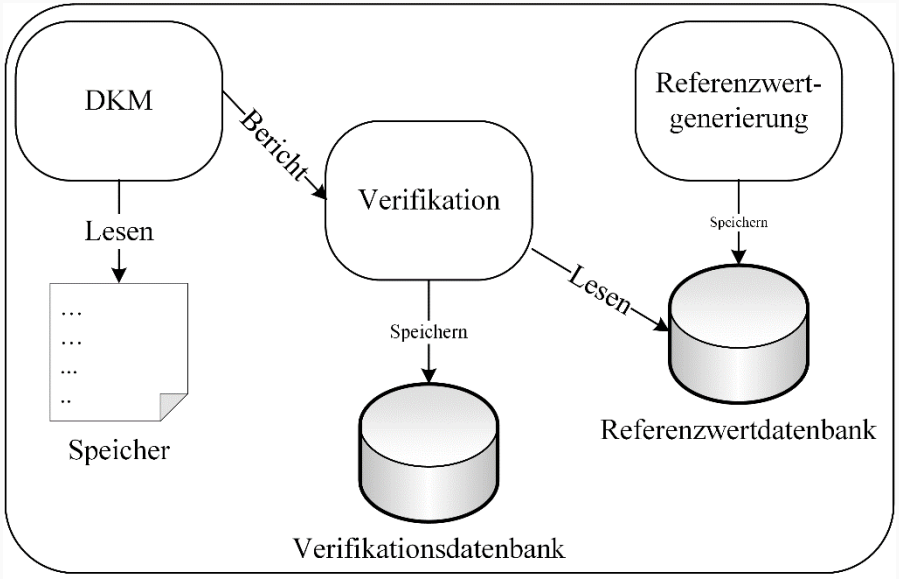
DECOIT®

Einleitung
Dynamische Integritätsmessung

- Ein besserer Ansatz ist die dynamische Integritätsmessung
- Messung der Integrität zur Laufzeit
  - Betriebssystem (Kernel + Loadable Kernel Module - LKM)
  - Prozesse
- Statische Speichersegmente werden gemessen:
  - Programmcode
  - Konstante Datensegmente
    - Konstanten (z.B.: `const int i = 42`)
    - String Literale (z.B.: `printf("string")`)
- Dynamische Elemente (z.B. im Stack) werden nicht erfassen
- Feststellung von Modifikationen im Arbeitsspeicher erfolgt daher nur im statischen Bereich

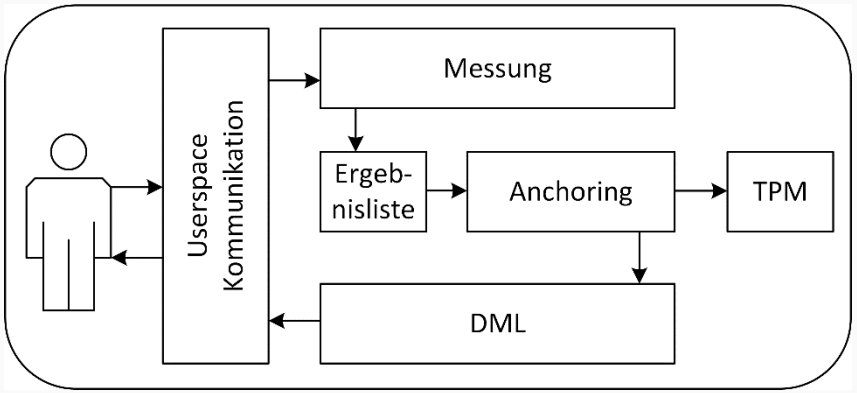
DRIVE Framework
Übersicht

- DRIVE besteht aus drei Komponenten (siehe Bild)
  - DRIVE Kernel Module (DKM): Messen des Kernelmoduls und System-State-Report-(SSR)Generator
  - Referenzwertgenerierung
  - Verifikation
- Verifikationsdatenbank
  - Speichern der Messungen und Ergebnisse
  - Für spätere Auswertung und Forensik



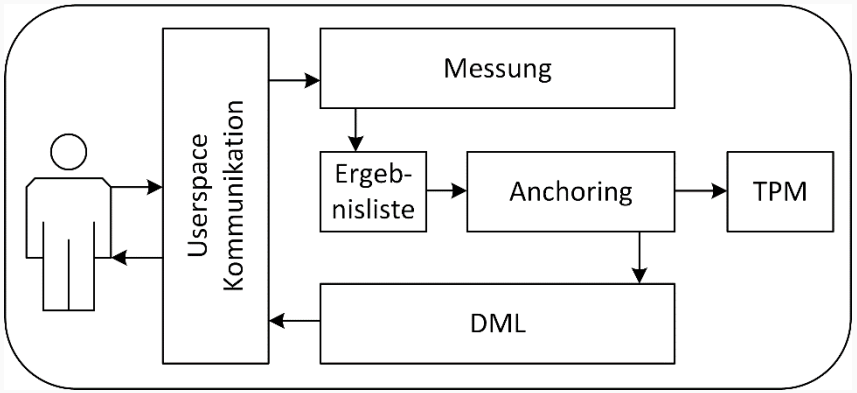
DRIVE Framework
DRIVE Kernel Module

- DRIVE Kernel Module (DKM) ist ein Kernelmodul für den Linux-Kernel: Loadable Kernel Module (LKM)
- Es beinhaltet ein Framework zur Messung von Userspace- und Kernelspace-Speicherbereichen
- Die Architektur kann in fünf Bereiche unterteilt werden (siehe Bild)
  - Userspace-Kommunikation
  - Messung
  - Anchoring
  - Trusted Platform Module (TPM)
  - Dynamic Measurement List (DML)



DRIVE Framework
Messung

- DRIVE Kernel Module (DKM)
  - Prüfsummenberechnungen der Speichersegmente
  - Erhebung von Metadaten (z.B. Page-Flags)
  - Ankern der Ergebnisse gegen einen TPM (Erkennen nachträglicher Modifikation)
  - Speichern der Messungen in DML
  - Read-Only-Liste, kann nur erweitert werden
  - Kein Entfernen von Einträgen der DML möglich
- Auslesen der Messungen durch SSR-Generator: System State Report
- Übertragung zum Verifikationssystem



DRIVE Framework
Referenzwertgenerierung

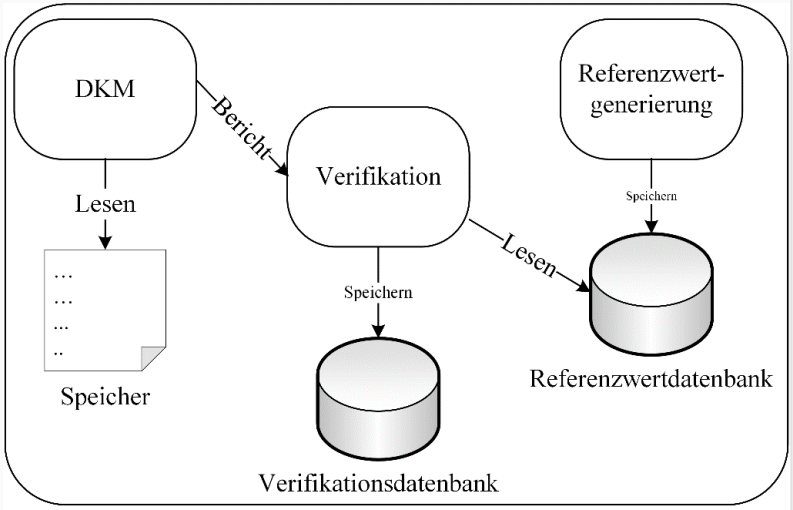
- Die Reference Value Generation (RVG) ist ein Gegenstück zum DKM: Framework zur Erzeugung von Referenzwerten
- Berechnung von Referenzwerten
  - Statische Konfigurationsdaten des Zielsystems
  - Nutzt vertrauenswürdige Quelldateien (ELF-Format), die denen auf dem Zielsystem entsprechen müssen
  - Konkrete Werte oder Prüfsummen
  - Statische Metadaten aus den ELF-Dateien: falls Nachberechnung durch Verifikation notwendig
  - Bereitgestellt über Referenzwertdatenbank

DRIVE Framework
Verifikation

- Decodieren und Prüfen des SSR-Werte
- Vergleich von Mess- und Referenzwerten
  - Direkter Vergleich von Werten oder Prüfsummen
  - Teilweise Nachberechnung notwendig
- Berechnung von Prüfsummen
  - Benötigt Laufzeitdaten aus dem Zielsystem
  - Benötigt statische Daten aus den Referenzwerten
  - Verrechnung dieser Daten mit bekannten Binärdaten
  - Berechnung der Prüfsumme aus den verrechneten Daten

Guidelines
Konzept

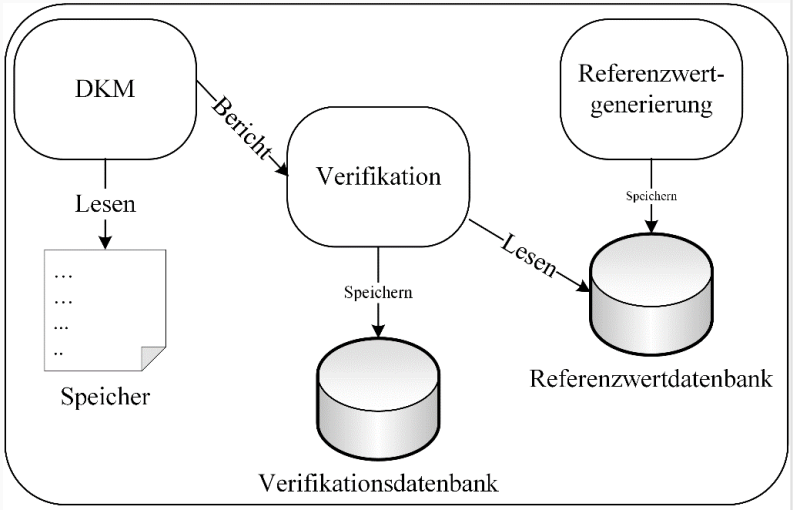
- Guidelines repräsentieren die notwendige Geschäftslogik, um konkrete Operationen auszuführen
- Logik für Messung und Verifikation ist modular aufgebaut
  - Kapselung
  - Vermeidung von Seiteneffekten
- Ein Guideline-Satz besteht immer aus drei Guidelines:
  - Messung
  - Referenzwertgenerierung
  - Verifikation
- Sätze können problemlos hinzugefügt und entfernt werden
- Sätze dürfen keine Abhängigkeiten untereinander haben



DRIVE Kernel Module (DKM) Framework

Guidelines
Beispiel

- DKM-Messung:
  - Auslesen der Speichersegmente, Hash-Berechnung
  - Auslesen von Page-Flags
  - Zählen der Pages eines Segments mit unerwarteten Flags (z.B. erwartet: r-x → gefunden: rwx)
- RVG-Referenzwertgenerierung:
  - Berechnung eines Referenz-Hash aus der ELF-Datei
  - Statische Konfiguration der erwarteten Page-Flags
- Verifikation
  - Vergleich von Hashes und Page-Flags (DKM-Messung mit RVG-Werten)
  - Prüfung des Zählers abweichender Pages auf 0 (valide)
  - Alle anderen Ergebnisse sind invalide



DRIVE Kernel Module (DKM) Framework

DECOIT®

Fazit

- Es wurde ein plattformunabhängiges und modulares Framework für sichere Messungen und Verifizierung von dynamischen Laufzeitinformationen unter Linux erarbeitet
- Dabei wurde auf dynamische Integritätsmessungen mittels TPM gesetzt
- Dadurch können Programmcode-Änderungen während der Laufzeit erkannt werden
- Dies wurde durch die Entwicklung eines Frameworks zur *Dynamic Runtime Attestation (DRA)* erreicht
- Die Attestierungsstrategie wurde durch Guidelines zentral umgesetzt
- Anpassungen auf neuere Kernelversionen sind zwar notwendig, aber nur noch reine Fleißarbeit
- Ein Demonstrator wurde zusätzlich entwickelt, um die Funktionalität widerzuspiegeln
- Dieser wurde auf dem TCG Members Meeting im Juli erfolgreich gezeigt

Vielen Dank für Ihre Aufmerksamkeit!

DECOIT GmbH
Fahrenheitstraße 9
D-28359 Bremen

https://www.decoit.de
info@decoit.de