NOSER ENGINEERING
WE KNOW HOW

Höhere Security durch frühe Entwurfsentscheidungen

Willi Flühmann

Embedded Computing Conference 2017

IT-Security in der Krise

Ein Milliardengeschäft?

- Regierungsfinanzierte Malware
- Gefährdete Infrastruktur
- Marktplätze im Darknet
- DDoS-Attacken
- Datenlecks
- Ransomware
- Phishing

IT-Security in der Krise

Ein Milliardengeschäft?

- Regierungsfinanzierte Malware
- Gefährdete Infrastruktur
- Marktplätze im Darknet
- DDoS-Attacken
- Datenlecks

Wieso?

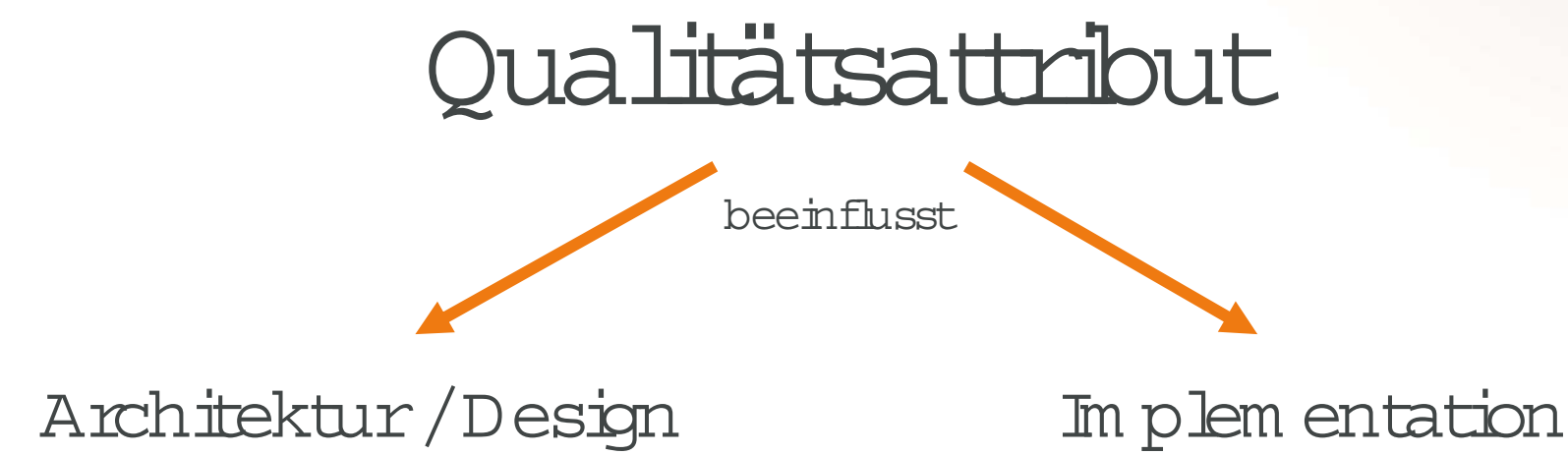
Die Welt wird immer vernetzter:
Internet der Dinge

Aktuell (Sommer 2017)

- Nach „WannaCry" folgt „SambaCry"
- Broadpwn: Kritische Lücke im WLAN-Chip vieler Smartphones
  → alle Geräte in Funkreichweite betroffen, keine Patches für alte Geräte
- Bug in RAR-Dekompression gefährdet Antivirus-Software:
  → Ausführung von fremdem Code beim Scan von RAR-Archiven

Was ist Security?

Nicht einfach ein Feature...



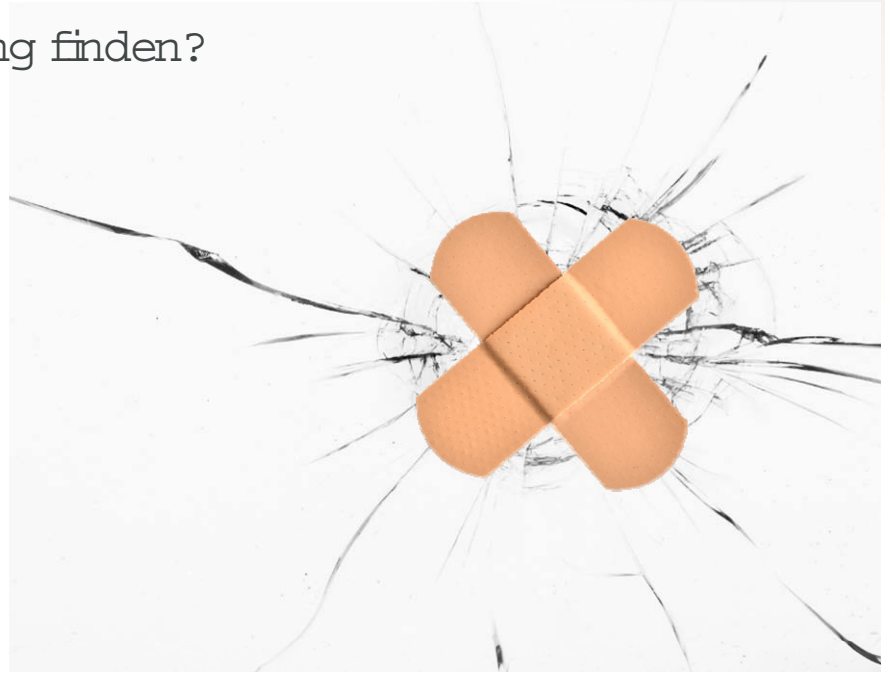
Weniger effektiv: Testing & Patching

- Besseres Testing kann schlampige Programmierung kaum kompensieren

  ➔ Sicherheitslücken während Testing finden?

- Embedded-Systeme werden selten aktualisiert

- Zero-Day-Exploits



Weniger effektiv: Testing & Patching

- Besseres Testing kann schlampige Programmierung kaum kompensieren

  ➔ Sicherheitslücken während Testing finden?

- Embedded-Systeme werden selten aktualisiert

Ein System kann unbemerkt kompromittiert worden sein...

Von Anfang an richtig machen

- Security-kritische Module:
  Korrektheit als vorrangiges Ziel

- Enge Beziehung zwischen
  „Security“ and „Safety“:

  Lassen wir uns bewährten Methoden
  bei der Safety inspirieren.

➔ Versagen ist in manchen Fällen nicht akzeptabel.

Wie beginnen wir?

CWE-Liste („Common Weakness Enumeration“)

ation Exposure Through Sent Data · Incorrect Behavior Order: Early Amplification · Improperly Implemented Security Check for Standard · Authorization Bypass Through User-Controlled Key · cious Comment · Information Exposure Through Server Log Files · Mismatched Memory Management Routines · Unexpected Sign Extension · Authentication Bypass: OpenSSL CTX Object Modified SSL Objects are Created · Insufficient Logging · Critical Variable Declared Public · Use of Incorrect Byte Ordering · Double-Checked Locking · Dangerous Signal Handler not Disabled During ive Operations · Improper Handling of Length Parameter Inconsistency · Authentication Bypass Issues · Execution with Unnecessary Privileges · Loop with Unreachable Exit Condition ('Infinite · Missing Critical Step in Authentication · Returning a Mutable Object to an Untrusted Caller · Unnecessary Complexity in Protection Mechanism (Not Using 'Economy of Mechanism') · Improper ing of Extra Values · Improper Neutralization of Substitution Characters · Information Exposure Through Behavioral Discrepancy · Incorrect Behavior Order · Improper Release of Memory Before ving Last Reference ('Memory Leak') · Sensitive Data Under Web Root · Stack-based Buffer Overflow · Incorrect Use of Privileged APIs · Improper Check or Handling of Exceptional Conditions · Seed in PRNG · Missing Required Cryptographic Step · Use of Password Hash With Insufficient Computational Effort · Improper Handling of Values · Failure to Handle Incomplete Element · ion Call with Incorrectly Specified Arguments · Use of Client-Side Authentication · External Initialization of Trusted Variables or Data Stores · Improper Control of Document Type Definition · n of Stack Variable Address · Improper Authorization in Handler for Custom URL Scheme · Improper Handling of Unexpected Data Type · Integer Underflow (Wrap or Wraparound) · Creation of orary File With Insecure Permissions · Reliance on Undefined, Unspecified, or Implementation-Defined Behavior · Use of Insufficiently Random Values · Improper Access Control · Improper Cross- lary Removal of Sensitive Data · Passing Mutable Objects to an Untrusted Method · Numeric Range Comparison Without Minimum Check · Improper Neutralization of Input Terminators · Return of Status Code · Unexpected Status Code or Return Value · External Control of File Name or Path · Use of Out-of-range Pointer Offset · Logging of Excessive Data · Improper Resolution of Path alence · Improper Handling of Windows Device Names · Information Exposure Through Debug Log Files · Information Exposure Through Externally-generated Error Message · Undefined Behavior put to API · Omitted Break Statement in Switch · External Control of Critical State Data · Improper Enforcement of Message Integrity During Transmission in a Communication Channel · ation of Catch for Generic Exception · Coding Standards Violation · Storage of Sensitive Data in a Mechanism without Access Control · Improper Neutralization of Special Elements used in an OS and ('OS Command Injection') · Predictable Seed in PRNG · Only Filtering One Instance of a Special Element · Incorrect Default Permissions · Use of Externally-Controlled Format String · retation Conflict · Use of Pointer Subtraction to Determine Size · Use of Incorrectly-Resolved Name or Reference · Information Exposure Through Cleanup Log Files · Predictability Problems · ation Exposure Through Timing Discrepancy · External Control of System or Configuration Setting · Assignment of a Fixed Address to a Pointer · Use of Dynamic Class Loading · File and Directory ation Exposure · Return of Pointer Value Outside of Expected Range · Unparsed Raw Web Content Delivery · Missing Release of File Descriptor or Handle after Effective Lifetime · Improper alization of Macro Symbols · Context Switching Race Condition · Plaintext Storage of a Password · Incorrect Behavior Order: Validate Before Canonicalize · Information Exposure Through Include e Code · Embedded Malicious Code · Information Exposure Through Shell Error Message · Call to Thread run() instead of start() · Improperly Controlled Modification of Dynamically-Determined t Attributes · Incomplete Cleanup · Unimplemented or Unsupported Feature in UI · Missing Release of Resource after Effective Lifetime · Comparing instead of Assigning · Inappropriate Encoding utput Context · Insufficient Session Expiration · Improper Handling of Unicode Encoding · Incomplete Filtering of One or More Instances of Special Elements · Improper Check for Dropped ges · Incorrect Check of Function Return Value · Violation of Secure Design Principles · Reliance on File Name or Extension of Externally-Supplied File · Misinterpretation of Input · Improper ol of Dynamically-Identified Variables · Missing Reference to Active Allocated Resource · Use of sizeof() on a Pointer Type · Improper Neutralization of Input During Web Page Generation ('Cross- cripting') · Improper Access of Indexable Resource ('Range Error') · Dangling Database Cursor ('Cursor Injection') · Authentication Bypass by Assumed-Immutable Data · Incorrect Authorization · l Public Variable Without Final Modifier · Unprotected Alternate Channel · Use of RSA Algorithm without OAEP · Key Exchange without Entity Authentication · Cleartext Storage of Sensitive ation in Memory · Improper Resource Locking · Improper Privilege Management · Empty Synchronized Block · Improper Verification of Intent by Broadcast Receiver · Small Space of Random s · Insecure Temporary File · Improper Authorization · Insufficient Control of Network Message Volume (Network Amplification) · Unprotected Transport of Credentials · Race Condition During s to Alternate Channel · Improper Validation of Function Hook Arguments · Only Filtering Special Elements Relative to a Marker · Exposed Unsafe ActiveX Method · Selection of Less-Secure thm During Negotiation ('Algorithm Downgrade') · Insufficient Verification of Data Authenticity · Inadequate Encryption Strength · Divide By Zero · Non-Replicating Malicious Code · Reversible One- ash · UNIX Hard Link · Absolute Path Traversal · Addition of Data Structure Sentinel · Privilege Dropping / Lowering Errors · The UI Performs the Wrong Action · Improper Neutralization of ent Delimiters · Direct Use of Unsafe JNI · Improper Following of a Certificate's Chain of Trust · Reliance on IP Address for Authentication · Spyware · Improper Enforcement of a Single, Unique

CWE-Liste („Common Weakness Enumeration")

ation Exposure Through Sent Data · Incorrect Behavior Order: Early Amplification · Improperly Implemented Security Check for Standard · Authorization Bypass Through User-Controlled Key · cious Comment · Information Exposure Through Server Log Files · Mismatched Memory Management Routines · Unexpected Sign Extension · Authentication Bypass: OpenSSL CTX Object Modified SSL Objects are Created · Insufficient Logging · Critical Variable Declared Public · Use of Incorrect Byte Ordering · Double-Checked Locking · Dangerous Signal Handler not Disabled During ive Operations · Improper Handling of Length Parameter Inconsistency · Authentication Bypass Issues · Execution with Unnecessary Privileges · Loop with Unreachable Exit Condition ('Infinite · Missing Critical Step in Authentication · Returning a Mutable Object to an Untrusted Caller · Unnecessary Complexity in Protection Mechanism (Not Using 'Economy of Mechanism') · Improper ing of Extra Values · Improper Neutralization of Sub[illegible] · Incorrect Behavior Order · Improper Release of Memory Before ving Last Reference ('Memory Leak') · Sensitive Dat[illegible] per Check or Handling of Exceptional Conditions · Seed in PRNG · Missing Required Cryptographic S[illegible] Values · Failure to Handle Incomplete Element · ion Call with Incorrectly Specified Arguments · Use[illegible] · Improper Control of Document Type Definition · of Stack Variable Address · Improper Authorization[illegible] ger Underflow (Wrap or Wraparound) · Creation of brary File With Insecure Permissions · Reliance on U[illegible] Values · Improper Access Control · Improper Cross- ary Removal of Sensitive Data · Passing Mutable Obj[illegible] per Neutralization of Input Terminators · Return of Status Code · Unexpected Status Code or Return V[illegible] g of Excessive Data · Improper Resolution of Path alence · Improper Handling of Windows Device Nam[illegible] ally-generated Error Message · Undefined Behavior put to API · Omitted Break Statement in Switch · [illegible] ing Transmission in a Communication Channel · ation of Catch for Generic Exception · Coding Standards Violation · Storage of Sensitive Data in a Mechanism without Access Control · Improper Neutralization of Special Elements used in an OS and ('OS Command Injection') · Predictable Seed in PRNG · Only Filtering One Instance of a Special Element · Incorrect Default Permissions · Use of Externally-Controlled Format String · retation Conflict · Use of Pointer Subtraction to Determine Size · Use of Incorrectly-Resolved Name or Reference · Information Exposure Through Cleanup Log Files · Predictability Problems · ation Exposure Through Timing Discrepancy · External Control of System or Configuration Setting · Assignment of a Fixed Address to a Pointer · Use of Dynamic Class Loading · File and Directory ation Exposure · Return of Pointer Value Outside of Expected Range · Unparsed Raw Web Content Delivery · Missing Release of File Descriptor or Handle after Effective Lifetime · Improper alization of Macro Symbols · Context Switching Race Condition · Plaintext Storage of a Password · Incorrect Behavior Order: Validate Before Canonicalize · Information Exposure Through Include e Code · Embedded Malicious Code · Information Exposure Through Shell Error Message · Call to Thread run() instead of start() · Improperly Controlled Modification of Dynamically-Determined t Attributes · Incomplete Cleanup · Unimplemented or Unsupported Feature in UI · Missing Release of Resource after Effective Lifetime · Comparing instead of Assigning · Inappropriate Encoding tput Context · Insufficient Session Expiration · Improper Handling of Unicode Encoding · Incomplete Filtering of One or More Instances of Special Elements · Improper Check for Dropped ges · Incorrect Check of Function Return Value · Violation of Secure Design Principles · Reliance on File Name or Extension of Externally-Supplied File · Misinterpretation of Input · Improper l of Dynamically-Identified Variables · Missing Reference to Active Allocated Resource · Use of sizeof() on a Pointer Type · Improper Neutralization of Input During Web Page Generation ('Cross- cripting') · Improper Access of Indexable Resource (Range Error') · Dangling Database Cursor ('Cursor Injection') · Authentication Bypass by Assumed-Immutable Data · Incorrect Authorization · l Public Variable Without Final Modifier · Unprotected Alternate Channel · Use of RSA Algorithm without OAEP · Key Exchange without Entity Authentication · Cleartext Storage of Sensitive ation in Memory · Improper Resource Locking · Improper Privilege Management · Empty Synchronized Block · Improper Verification of Intent by Broadcast Receiver · Small Space of Random s · Insecure Temporary File · Improper Authorization · Insufficient Control of Network Message Volume (Network Amplification) · Unprotected Transport of Credentials · Race Condition During s to Alternate Channel · Improper Validation of Function Hook Arguments · Only Filtering Special Elements Relative to a Marker · Exposed Unsafe ActiveX Method · Selection of Less-Secure thm During Negotiation ('Algorithm Downgrade') · Insufficient Verification of Data Authenticity · Inadequate Encryption Strength · Divide By Zero · Non-Replicating Malicious Code · Reversible One- lash · UNIX Hard Link · Absolute Path Traversal · Addition of Data Structure Sentinel · Privilege Dropping / Lowering Errors · The UI Performs the Wrong Action · Improper Neutralization of ent Delimiters · Direct Use of Unsafe JNI · Improper Following of a Certificate's Chain of Trust · Reliance on IP Address for Authentication · Spyware · Improper Enforcement of a Single, Unique

Hand aufs Herz:
Kann man sich das alles merken?

Besser: Ursachen angehen

Design

Horizontale Partitionierung

Vertikal Partitionierung

Kritische Code-Teile klein halten

Security-Anforderungen an Plattform weitergeben

Kein wahlfreier Zugriff auf alle Objekte

Einbezug des Benutzers

...

Implementation

Typsicherheit

Sichere Verarbeitung von Strings

Definition spezialisierter Typen um Missbrauch zu vermeiden

Abstraktionen definieren und verarbeiten

Pointer-freie Programming (C++)

Wrapping von rohen Buffern (C++)

...

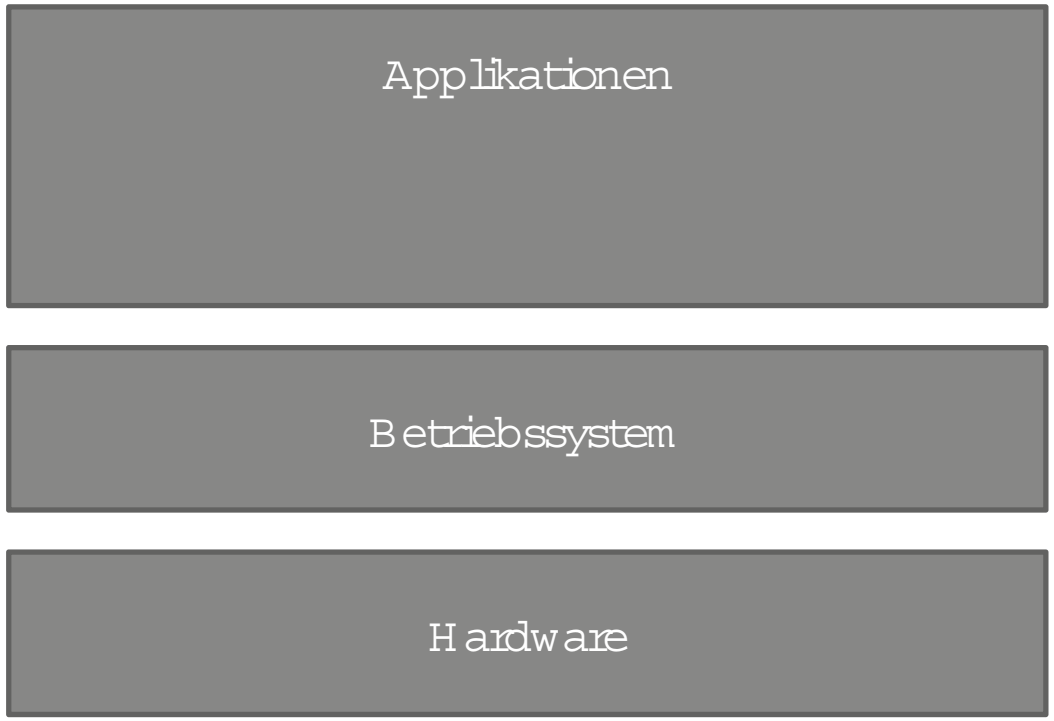
Horizontale Partitionierung

Prozesse sind im Speicher voneinander isoliert.

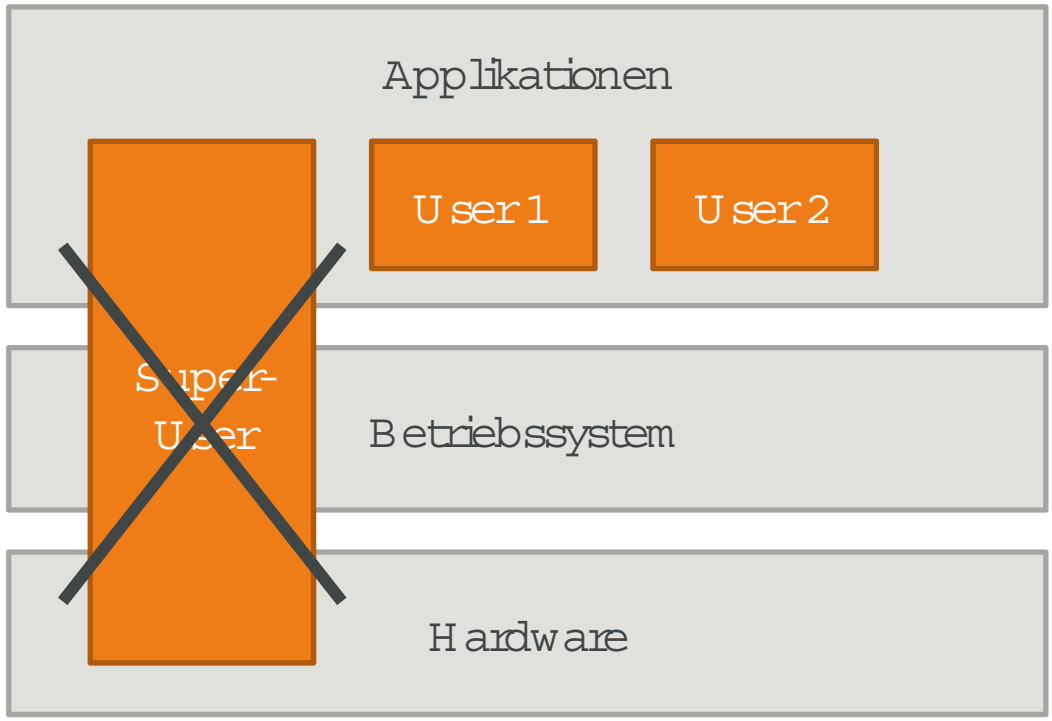
Aber was ist mit:

- Dateisystem / Logs
- Registry-Einträgen
- Zwischenablage
- Bildschirminhalt
- ...

Vertikale Partitionierung



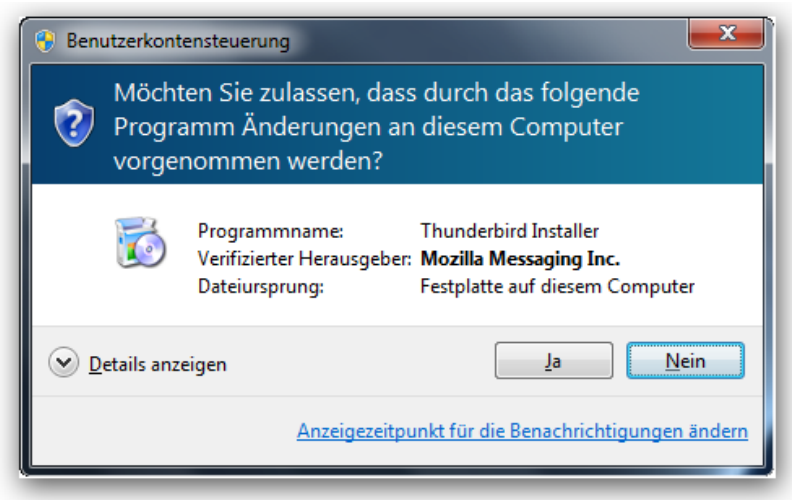
Vertikale Partitionierung



Applikationscode strikt trennen von „Super-User"-Fähigkeiten:

➔ Risiko, dass die Security-Infrastruktur unterlaufen wird.

Vertikale Partitionierung (Beispiel)



Dies bedeutet im schlimmsten Fall:

Möchten Sie diesem Programm ein unwiderrufliches Recht gewähren, unlimitierte und fortlaufende Änderungen zu machen inkl. Installation von nicht feststellbarer Malware.

Wenn das Programm aus einer zweifelhaften Quelle kommt, kann diesem Computer nie mehr vertraut werden.

Vertikale Partitionierung

Idee:

Low-Level-Systemmanipulationen sollen nur in einem speziellen Wartungsmodus verfügbar sein.

Ein Wechsel in diesen Modus erfordert eine fälschungssichere Benutzeraktion wie das Drücken einer physischen Taste oder das Auslösen eines Resets.

Kritische Code-Teile klein halten

„Complexity is the enemy of security."

Achtung!

Ohne saubere Isolation muss der Rest der Software ebenfalls als kritisch betrachtet werden.

(inkl. Betriebssystem und Bibliotheken)

Umgang mit kritischem Code

Make

Selbstgeschriebener Code hat häufig mehr Bugs.

Buy

Besser geprüft and gewartet?

Umgang mit kritischem Code

Make

Selbstgeschriebener Code hat häufig mehr Bugs.

Buy

Besser geprüft and gewartet?

Wenn eine gekaufte Lösung überladen ist, dann sie gesamthaft ebenfalls viele Bugs enthalten.
➔ nicht immer sicherere Weg

Durchschnittliche Lebensdauer eines Bugs im Linux-Kernel: 5 Jahre

Security-Anforderungen an die Plattform weitergeben

Herausforderungen:

- Verarbeitung heikler Daten
- Bedarf an vertrauenswürdigen Informationen vom Netzwerk oder der Peripherie

Eine Applikation kann diese Probleme nicht alleine lösen:

Die Plattform muss die Art der Daten kennen und Unterstützung bieten.

Security-Anforderungen an die Plattform weitergeben

Verantwortlichkeiten der Plattform :

Kein wahlfreier Zugriff auf alle Objekte

Problem:

Nach einer Autorisierung kann eine Software-Komponente normalerweise auf alle Objekte des gleichen Typs zugreifen.

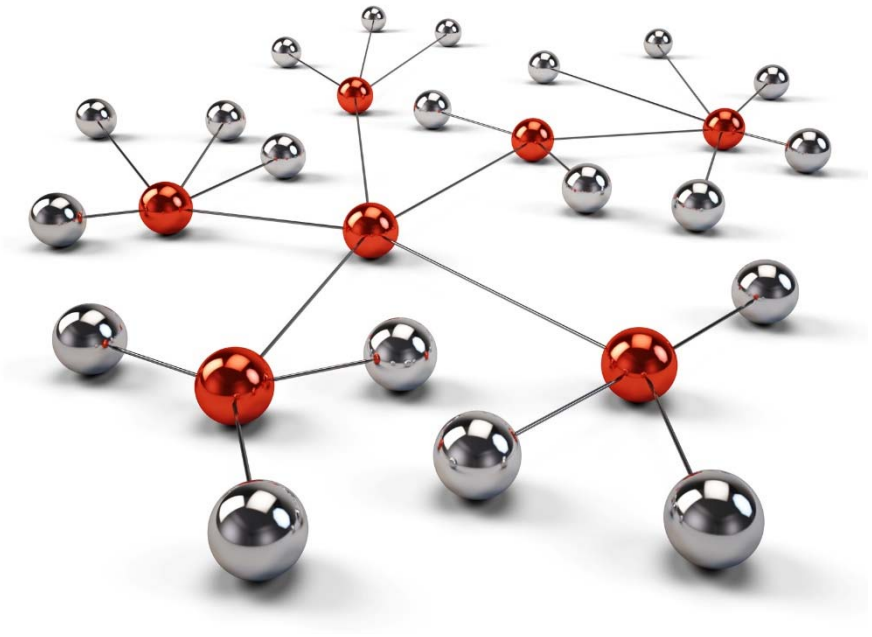
➔ nur durch Kenntnis eines Namens oder einer ID.

Kein wahlfreier Zugriff auf alle Objekte

Beispiele:

- Datensätze in einer Datenbank
- Dateien in einem Dateisystem
- Hosts in einem Netzwerk, Ports auf einem Host
- Seiten auf einem Web-Server

Kein wahlfreier Zugriff auf alle Objekte



Besser:

Autorisierung bezieht sich nur auf bestimmte Objekte („Kenntnis nur wenn nötig").

Auffinden anderer Objekte nur entlang vordefinierter Beziehungen.

Einbezug des Benutzers

Transparenz:

(Verdächtige) Hintergrundaktivitäten für den Benutzer sichtbar machen.

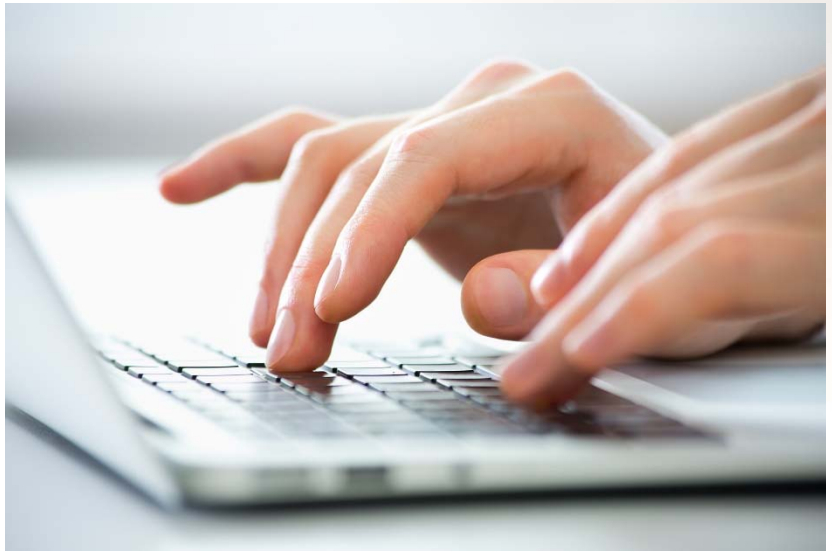
Überwachung des System zustands für den Benutzer erleichtern:

- „Root of Trust" zur Verfügung stellen
  (z.B. Möglichkeit ab einem anderen Medium zu booten)
- System zustand zusammenfassen
  (z.B. installierte Anwendungen und Änderungen)

Implementation

Im Allgemeinen:

Sprachen und Methoden bevorzugen, welche die Korrektheit von an Anfang gewährleisten.



Implementation

- Typsicherheit
- Sichere Verarbeitung von Strings
- Definition spezialisierter Typen um Missbrauch zu vermeiden
- Abstraktionen

Implementation

- Typsicherheit
- Sichere Verarbeitung von Strings
- Definition spezialisierter Typen um Missbrauch zu vermeiden
- Abstraktionen

Buffer-Overflows, Datenkorruption, ungültige Pointer, etc.

Implementation

- Typsicherheit
- Sichere Verarbeitung von Strings
- Definition spezialisierter Typen um Missbrauch zu vermeiden
- Abstraktionen

Explizite Stringtypen für:

- Datenstrings
- SQL/Kommando-Fragmente
- usw.

Umwandlung dazwischen bildet in einen sicheren Zeichensatz ab („Escaping").

Implementation

- Typsicherheit
- Sichere Verarbeitung von Strings
- Definition spezialisierter Typen um Missbrauch zu vermeiden
- Abstraktionen

Typsystem für die eigene Domäne etablieren.

Implementation

- Typsicherheit
- Sichere Verarbeitung von Strings
- Definition spezialisierter Typen um Missbrauch zu vermeiden
- Abstraktionen

Ein Programmierer sollte primär mit Abstraktionen arbeiten:

Definieren und nachfolgendes Verwenden.

Implementation (C++)

- „Pointer-freie" Programmierung
- Wrapping von rohen Buffern

→ Diskussion

NOSER ENGINEERING
WE KNOW HOW

Vielen Dank

für Ihre Aufmerksamkeit!

willi.fluehmann@noser.com

Twitter: @wfluehmann

NOSER ENGINEERING AG WINTERTHUR | LUZERN | BERN | MÜNCHEN | HEPPENHEIM

RUDOLF-DIESEL-STRASSE 3
CH-8404 WINTERTHUR
TEL +41 52 234 56 11

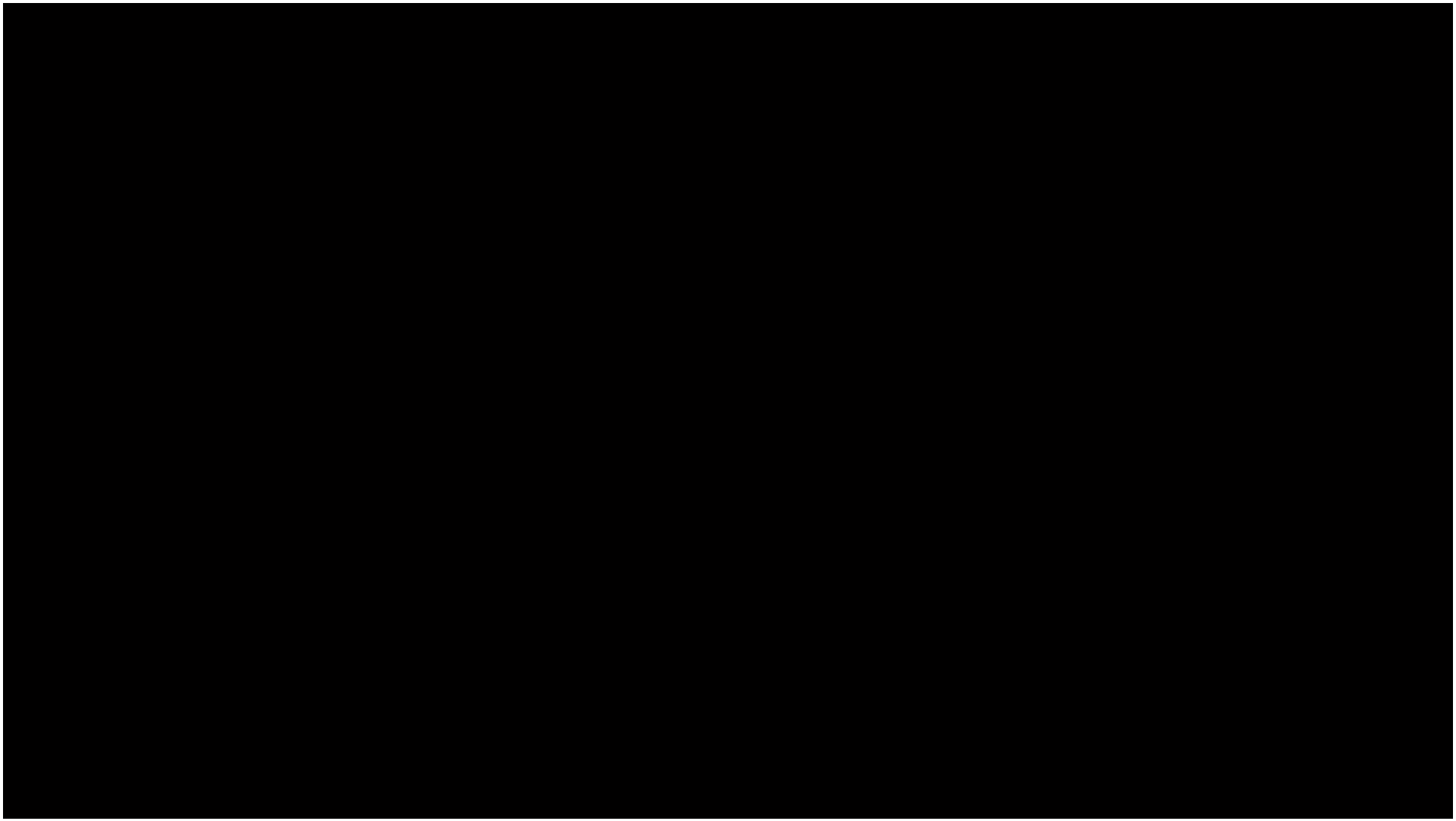
PLATZ 4
CH-6039 ROOT D4
TEL +41 41 455 66 11

GALGENFELDWEG 18
CH-3006 BERN
TEL +41 31 917 45 11

KONRAD-ZUSE-PLATZ 1
DE-81829 MÜNCHEN
TEL +49 89 9901 4880

DONNERSBERGSTRASSE 1
DE-64646 HEPPENHEIM
TEL +49 62 5267 4450

WWW.NOSER.COM | INFO@NOSER.COM



„Normale" Systeme vs. Embedded-Systeme

Normal

Standard-Komponenten wählen und regelmässige Updates

Embedded

Massgeschneiderte Komponenten hoher Qualität und auf ein Minimum abgespeckt

Typen von Strings

- `rawString = readLineFromFile()`

- `sqlFragment = „select * from users where id = "`
- `dataString = „123"`
- `sqlStatement = sqlFragment + dataString`

- `password = „mypassword"`

Typen von Strings

- `rawString = readLineFromFile()`

- `sqlFragment = „select * from users where id = "`
- `dataString = „123"`
- `sqlStatement = sqlFragment + dataString`

- `password = „mypass`

SQL-Fragment und Daten-String als separate Typen?

→ Umwandlung des Daten-Strings vor der Verkettung nötig, macht ein Escaping

Typen von Strings

- rawString = readLineFr

Kompiler und Betriebssystem sollten wissen, dass hier heikle Daten gespeichert werden

- sqlFragment = „select * from users where id = "
- dataString = „123"
- sqlStatement = sqlFragment + dataString

- password = „mypassword"

Ist ein Byte immer ein Byte?

1. Rohe Bytes von einer Datei oder aus einem Kommunikationsprotokoll
2. Eine Zahl
3. Ein Zeichen
4. Ein Boolean-Wert

Fragen:

- Bei welchen Typen sollen arithmetische Operationen (plus, minus, …) oder bitweise Operationen erlaubt sein?
- Bei welchen Typen sollen Integer-Overflows ausgeschlossen werden?

Pointer-freie Programmierung in C++ (1)

Typische Verwendung von Pointern:

- Polymorphe Objekte
- Objekte mit Inhalt variabler Grösse
- Container für eine variable Anzahl Elemente
- Factorymethode gibt Objekte zurück
- Optionale Objekte
- Exzessives Kopieren vermeiden bei der Übergabe

Pointer-freie Programmierung in C++ (2)

Pointer-freie Alternativen:

- C++-Container (std::vector, usw.)
- std::unique_ptr
- Als Wert übergeben und als Wert zurückgeben
  (muss nicht teuer sein dank Move-Konstruktoren und „Copy-Elision“)
- boost::optional, std::optional (C++17)

Diskussion

- Stehen wir bezüglich Cyberangriffen noch ganz am Anfang?
- Können wir unsere Systeme mit ein paar Handgriffen sicher machen oder sind sie grundsätzlich nicht mehr vertrauenswürdig?
- Welche Programmiersprache eignet sich für besonders sicheren Code?