Implementierungsframework für die Schadsoftwareerkennung auf Android

Theodor Nolte (theodor.nolte@haw-hamburg.de)

Hochschule für Angewandte Wissenschaften Hamburg
Fakultät Technik und Informatik
Department Informatik

11. Mai 2011

university-logo

Gliederung

university-logo

SKIMS

Bundesministerium
für Bildung
und Forschung

***S**chichtenübergreifendes **k**ooperatives **I**mmunsystem*
*für **m**obile, mehrseitige **S**icherheit*

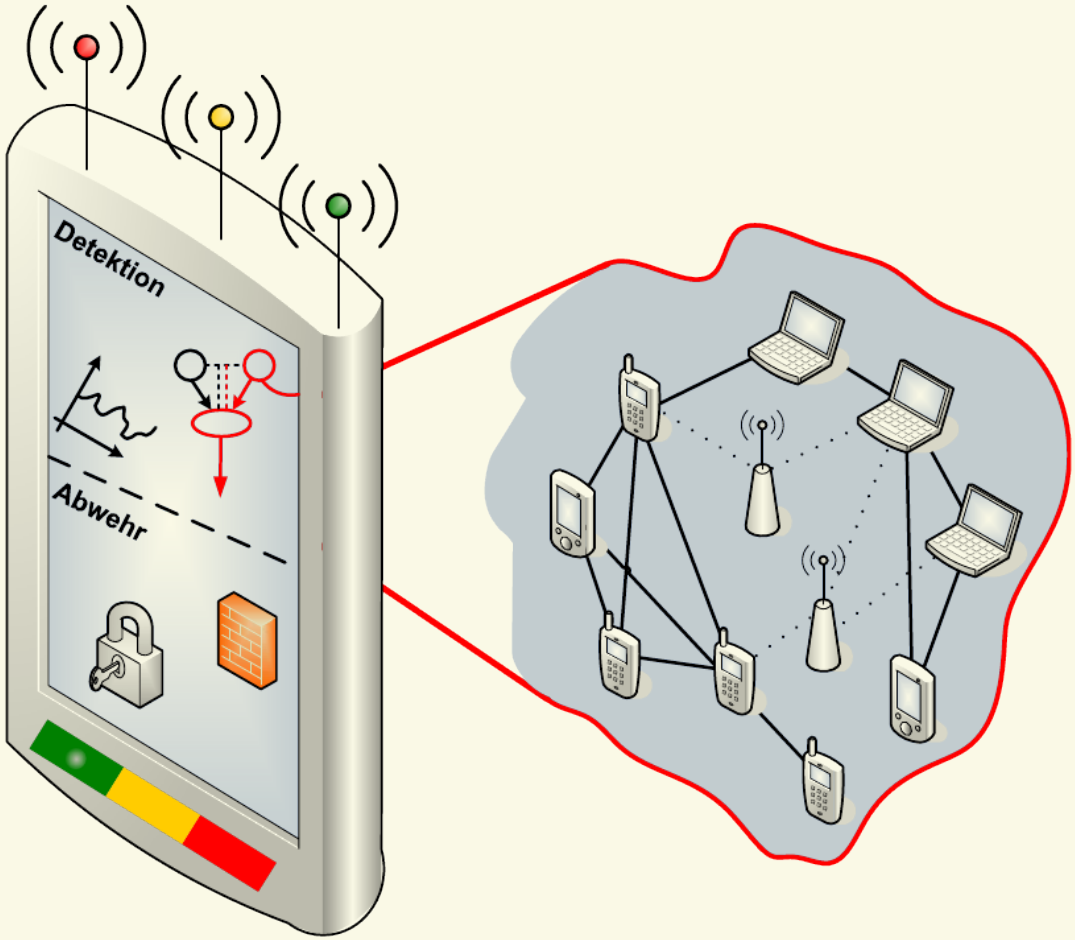
Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

university-logo

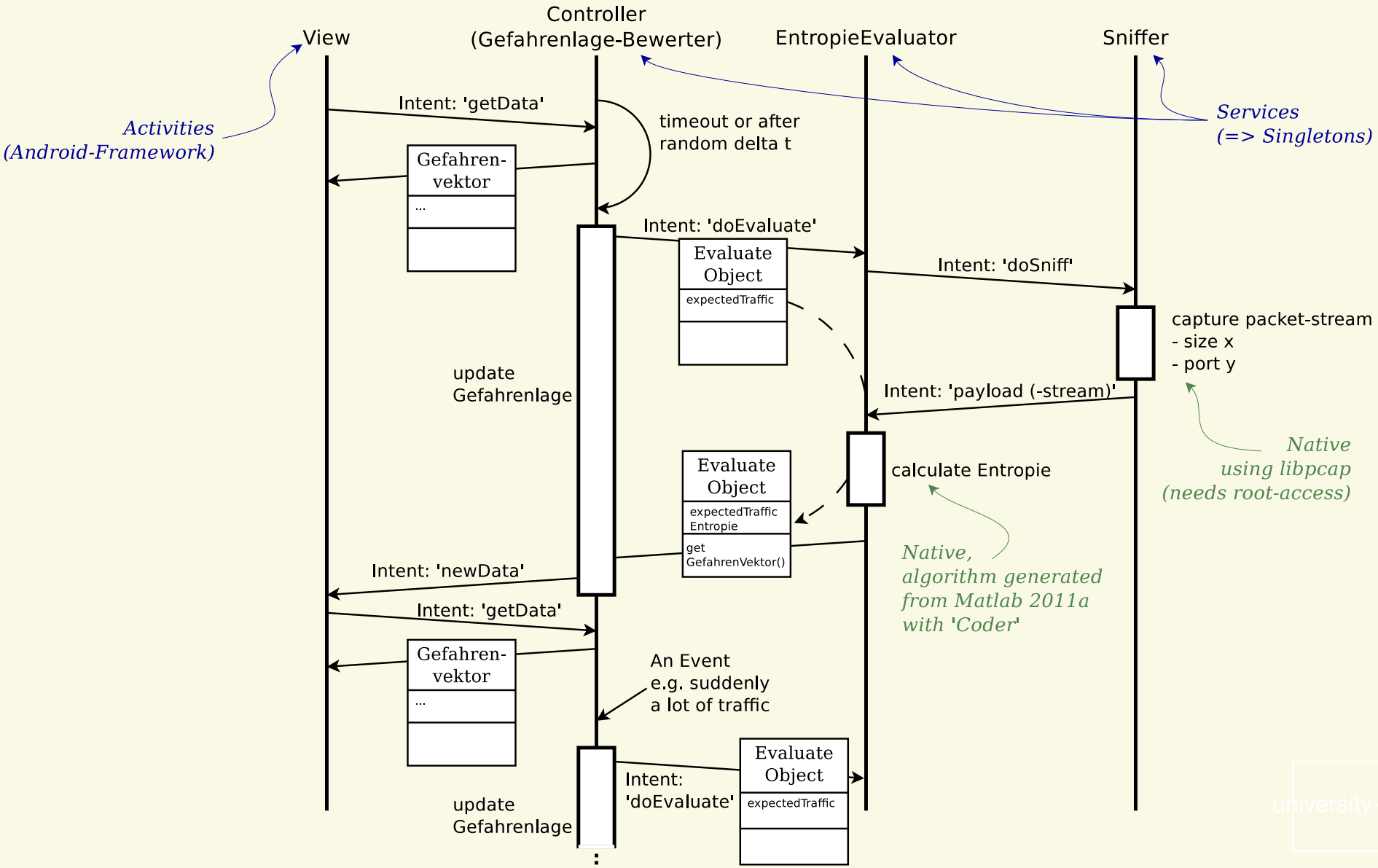
SKIMS

Angriffe

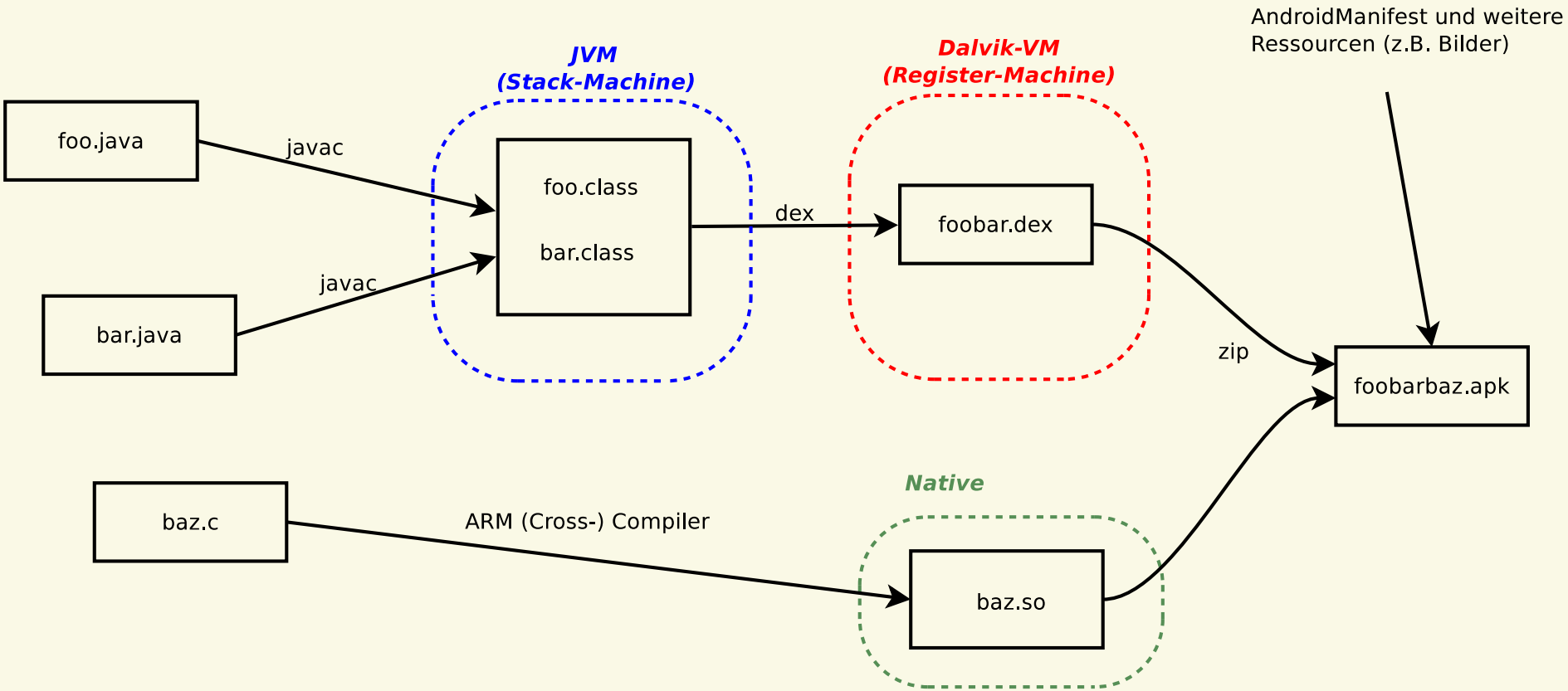
- erkennen
- abwehren



Sequenzdiagramm



Vom SourceCode zur App (.apk)

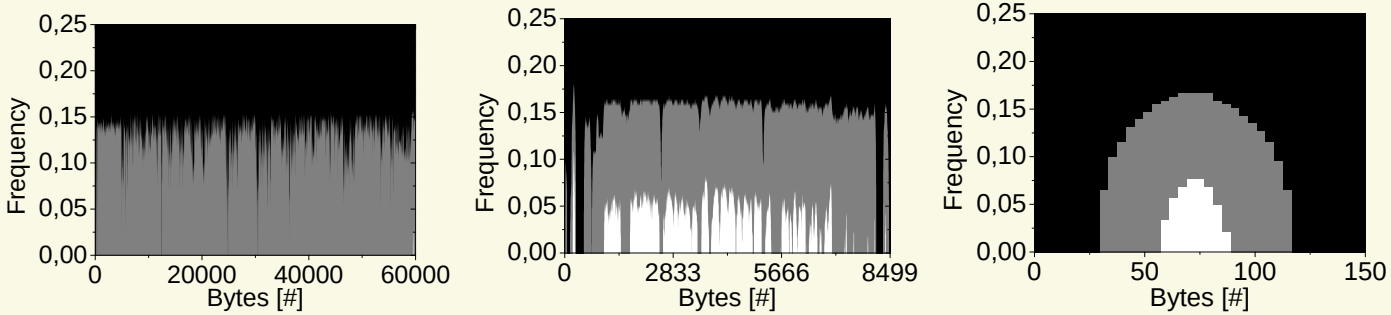


Entropie-Analyse

Entropie: Maß für den Informationsgehalt

$H(x) = -\sum_{i=1}^{n} p(X_i) \cdot \log_2 p(X_i)$ [6]

Zuständig: Benjamin Jochheim



[4]

(a) Text (b) Binary (c) Shellcode

Gliederung

university-logo

AASandbox

Technische Universität Berlin (DAI-Labor)

Android Application Sandbox System for Suspicious Software Detection [2]

Analyse von Apps

- statisch: Analyse der .app-Dateien
- dynamisch: App wird innerhalb der Sandbox ausgeführt

university-logo

AASandbox

Statische Analyse

Signaturüberprüfung

Erkennen charakteristischer **Codestruktur**: Shellcode (malicious code fragments)

- schnell
- relativ einfach
- automatisch

AASandbox

Dynamische Analyse

Sandbox

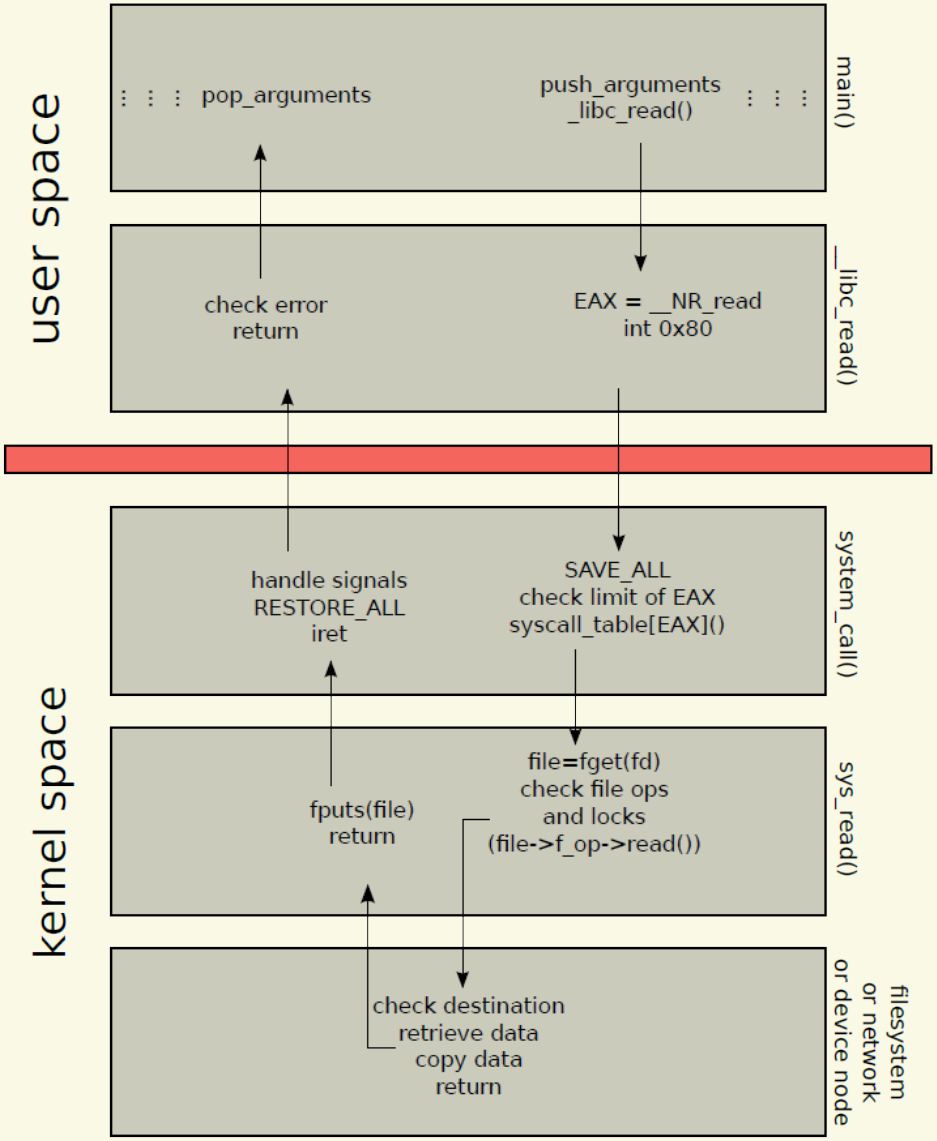
> *. . . an environment in which the actions of a process are restricted according to a security policy.*

[1]

Erkennen von charakteristischem **Verhalten**: Dateioperationen, Netzwerkaktivität, Prozesse u. Threads, System-Calls

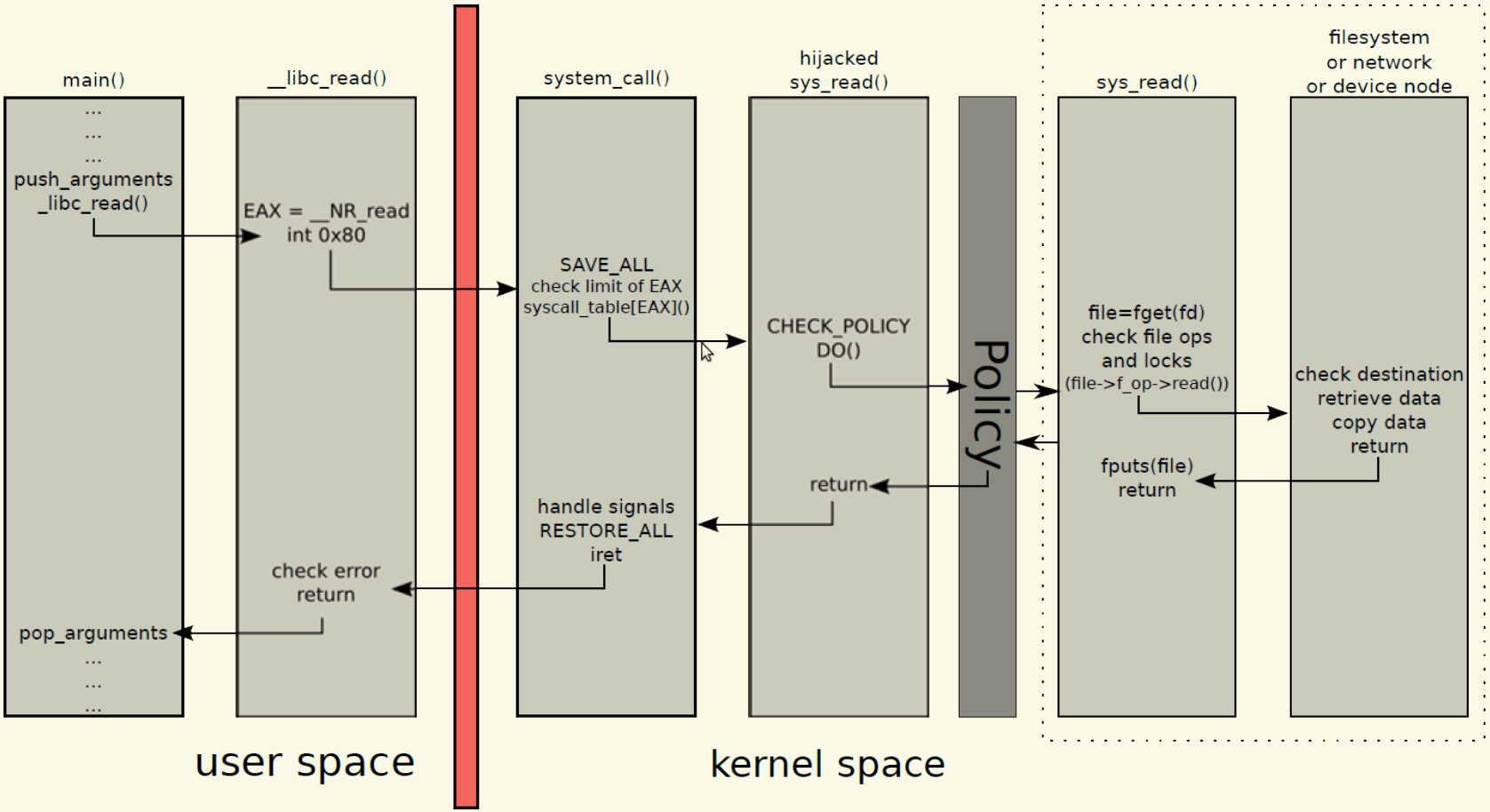
- langwierig
- komplex
- manuell und automatisch

AASandbox


user space
kernel space
pop_arguments
push_arguments
_libc_read()
main()
check error
return
EAX = __NR_read
int 0x80
__libc_read()
handle signals
RESTORE_ALL
iret
SAVE_ALL
check limit of EAX
syscall_table[EAX]()
system_call()
fputs(file)
return
file=fget(fd)
check file ops
and locks
(file->f_op->read())
sys_read()
check destination
retrieve data
copy data
return
filesystem
or network
or device node

[2]

AASandbox


main()
...
...
...
push_arguments
_libc_read()
pop_arguments
...
...
...
__libc_read()
EAX = __NR_read
int 0x80
check error
return
system_call()
SAVE_ALL
check limit of EAX
syscall_table[EAX]()
handle signals
RESTORE_ALL
iret
hijacked
sys_read()
CHECK_POLICY
DO()
return
Policy
sys_read()
file=fget(fd)
check file ops
and locks
(file->f_op->read())
fputs(file)
return
filesystem
or network
or device node
check destination
retrieve data
copy data
return
user space
kernel space

[2]

AASandbox – Abgrenzung

nicht mobil (ab in die Cloud)

statisch und dynamisch, aber nicht kontinuierlich

tief, systemnah (Analyse)

Zero-day Exploits

Gliederung

university-logo

KBTA Method

Ben-Gurion University, Israel

Knowledge-Based, Temporal Abstraction (KBTA) Method [5]

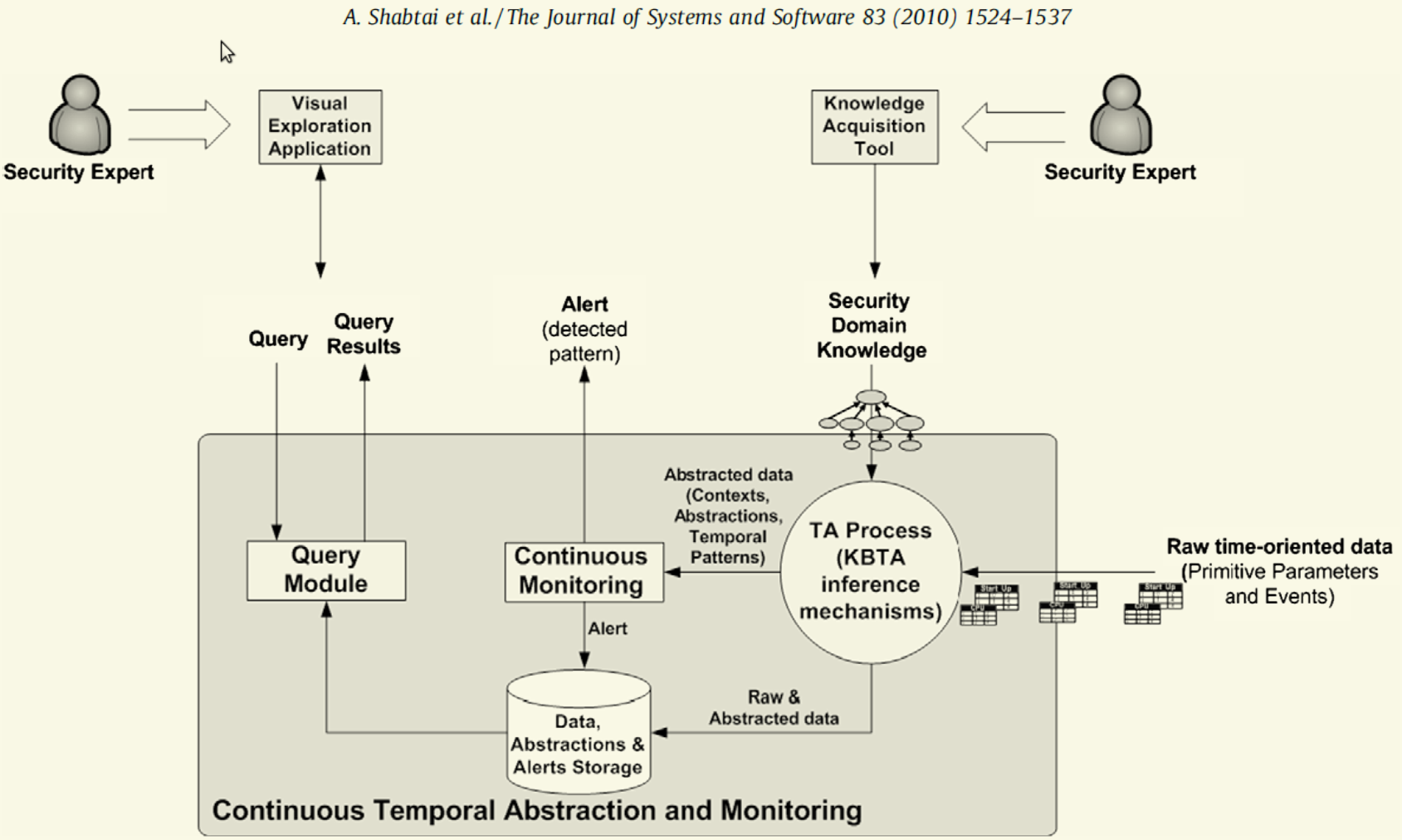
Vergleich von kontinuierlichen, abstrakten Ereignissen mit Wissensbasis, einer **Ontologie**

Ereignisse:

- CPU-Auslastung
- Traffic-Rate
- Benutzerinteraktionen
- ...

university-logo

KBTA


A. Shabtai et al. / The Journal of Systems and Software 83 (2010) 1524–1537
Security Expert
Visual Exploration Application
Knowledge Acquisition Tool
Security Expert
Query
Query Results
Alert (detected pattern)
Security Domain Knowledge
Abstracted data (Contexts, Abstractions, Temporal Patterns)
TA Process (KBTA inference mechanisms)
Raw time-oriented data (Primitive Parameters and Events)
Query Module
Continuous Monitoring
Alert
Raw & Abstracted data
Data, Abstractions & Alerts Storage
Continuous Temporal Abstraction and Monitoring

[5]

KBTA – Abgrenzung

kontinuierliche Verfolgung von Ereignissen

für ein *echtes* Android-Gerät: optimiert auf geringen Ressourcenverbrauch

möglicherweise Sicherheitsproblem (Informationen in Datenbank)

Zero-day Exploits?

university-logo

Gliederung

university-logo

honeyM

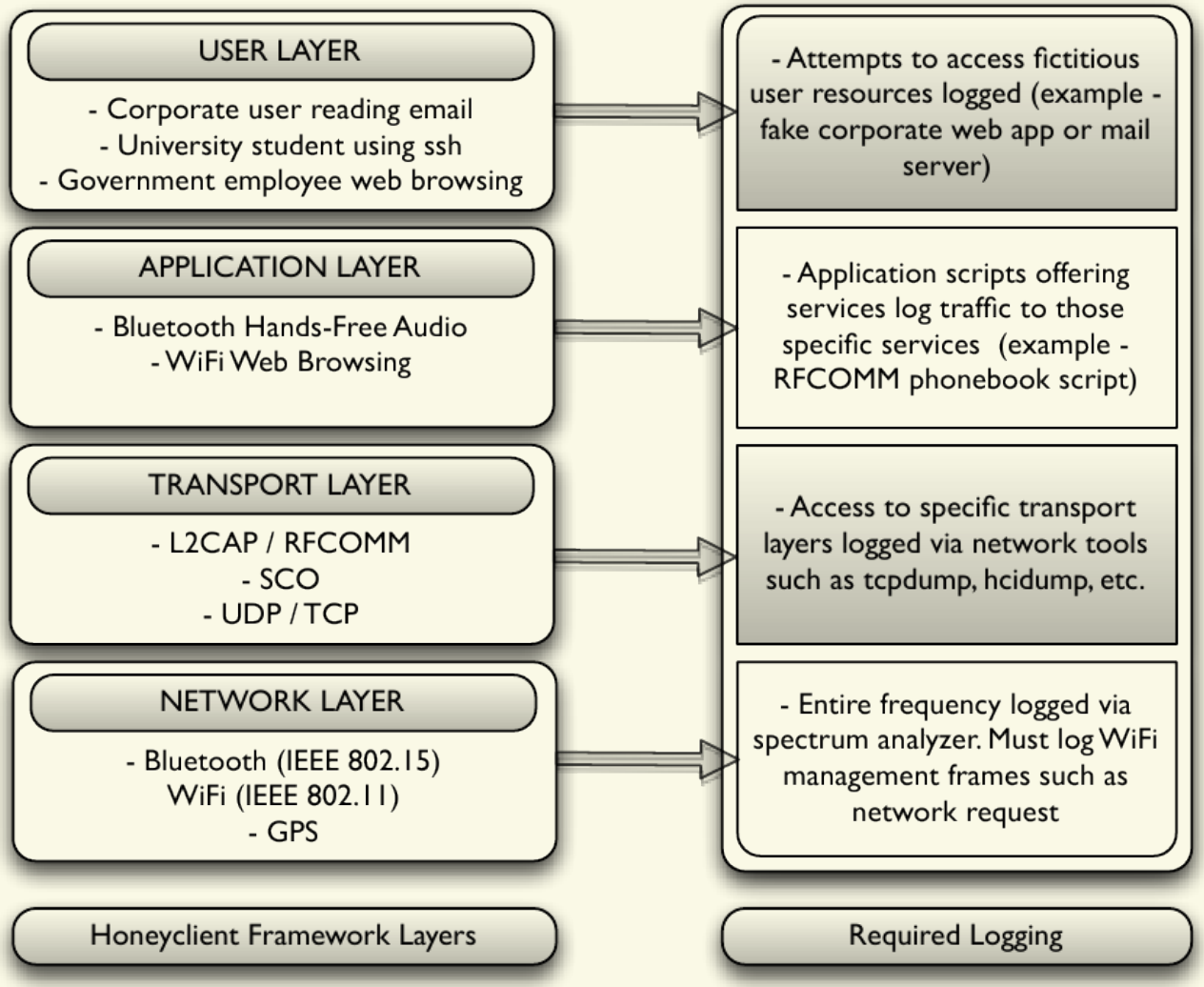
US Military Academy, Westpoint NY

A Framework for Implementing Virtual Honeyclients for Mobile Devices [3]

Rechner simuliert mehrere Mobilgeräte (z.B. IPhones)

- simuliert WLAN, Bluetooth und GPS
  - WLAN
  - Bluetooth
  - GPS
- Fingerprinting

KBTA


USER LAYER
- Corporate user reading email
- University student using ssh
- Government employee web browsing
APPLICATION LAYER
- Bluetooth Hands-Free Audio
- WiFi Web Browsing
TRANSPORT LAYER
- L2CAP / RFCOMM
- SCO
- UDP / TCP
NETWORK LAYER
- Bluetooth (IEEE 802.15)
WiFi (IEEE 802.11)
- GPS
Honeyclient Framework Layers
- Attempts to access fictitious user resources logged (example - fake corporate web app or mail server)
- Application scripts offering services log traffic to those specific services (example - RFCOMM phonebook script)
- Access to specific transport layers logged via network tools such as tcpdump, hcidump, etc.
- Entire frequency logged via spectrum analyzer. Must log WiFi management frames such as network request
Required Logging

[3]

honeyM – Abgrenzung

stationärer Rechner => nicht mobil (und keine Cloud)

umfangreich (Interfaces)

aufwendig (Fingerprinting)

Zero-day Exploits

Zusammenfassung

Entropie-Analyse – Eigener Ansatz [4]

Sandbox – AASandbox [2]

Ontologie – KBTA [5]

Honeypot – honeyM [3]

university-logo

Vielen Dank für Eure Aufmerksamkeit

university-logo

Literatur I

[1] Matthew A. Bishop.
The Art and Science of Computer Security.
Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[2] Thomas Bläsing, Aubrey-Derrick Schmidt, Leonid Batyuk, Seyit A. Camtepe, and Sahin Albayrak.
An android application sandbox system for suspicious software detection.
In 5th International Conference on Malicious and Unwanted Software (Malware 2010), Nancy, France, France, 2010.

[3] T. J. O'Connor and Ben Sangster.
honeym: a framework for implementing virtual honeyclients for mobile devices.
In Proceedings of the third ACM conference on Wireless network security, WiSec '10, pages 129–138, New York, NY, USA, 2010. ACM.

[4] Thomas C. Schmidt, Matthias Wählisch, and Michael Gröning.
Context-adaptive Entropy Analysis as a Lightweight Detector of Zero-day Shellcode Intrusion for Mobiles.
In Proc. of the ACM WiSec, New York, June 2011. ACM.
Accepted for poster presentation.

[5] Asaf Shabtai, Uri Kanonov, and Yuval Elovici.
Intrusion detection for mobile devices using the knowledge-based, temporal abstraction method.
Journal of Systems and Software, 83(8):1524 – 1537, 2010.
Performance Evaluation and Optimization of Ubiquitous Computing and Networked Systems.

[6] C. E. Shannon.
A Mathematical Theory of Communication.
Bell System Technical Journal, 27:379–423, 623–656, July, October 1948.

Application

Eine **Application** setzt sich aus den **Components** zusammen

Components:

- **Activity**
- **Service**
- **BroadcastReceiver**
- **ContentProvider**

Sowie dem **AndroidManifest.xml**

und den Ressourcen, z.B. Bilder (Unterordner **res**/)

university-logo

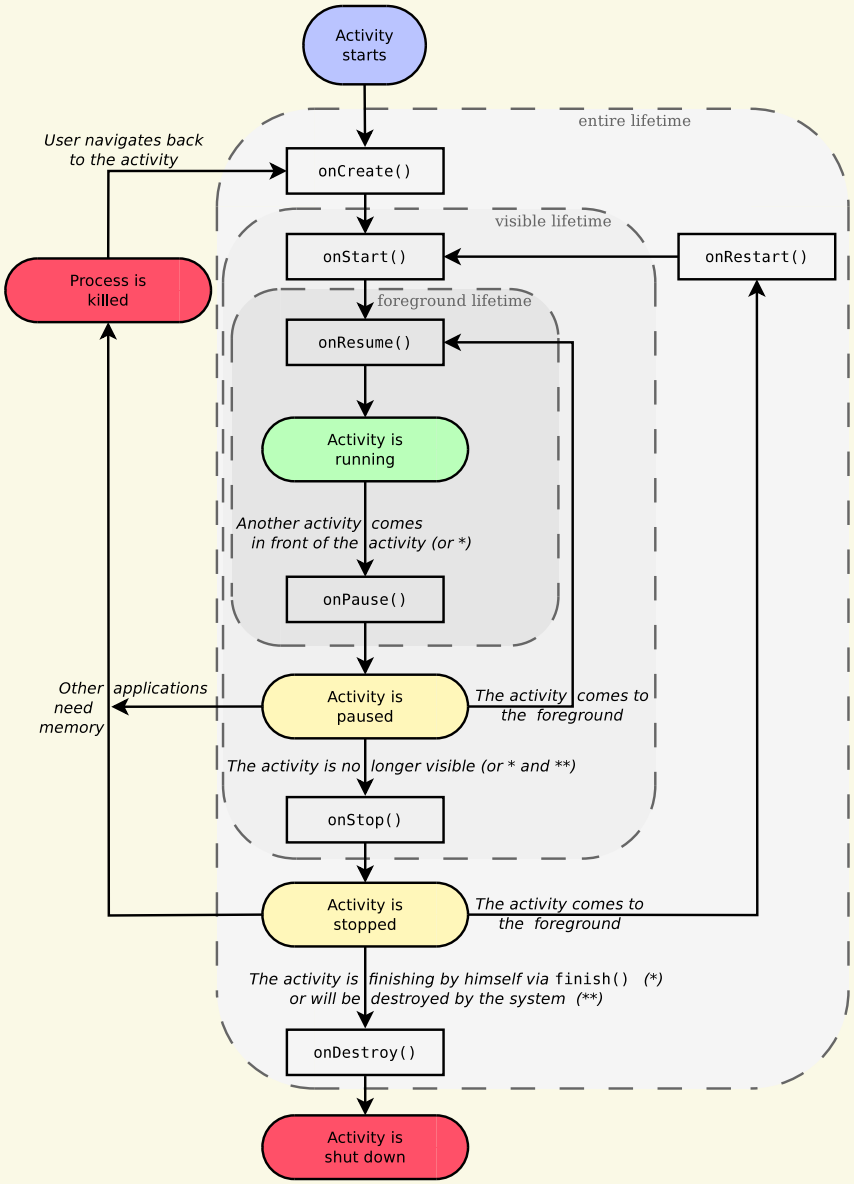
Application

Keine Main-Methode

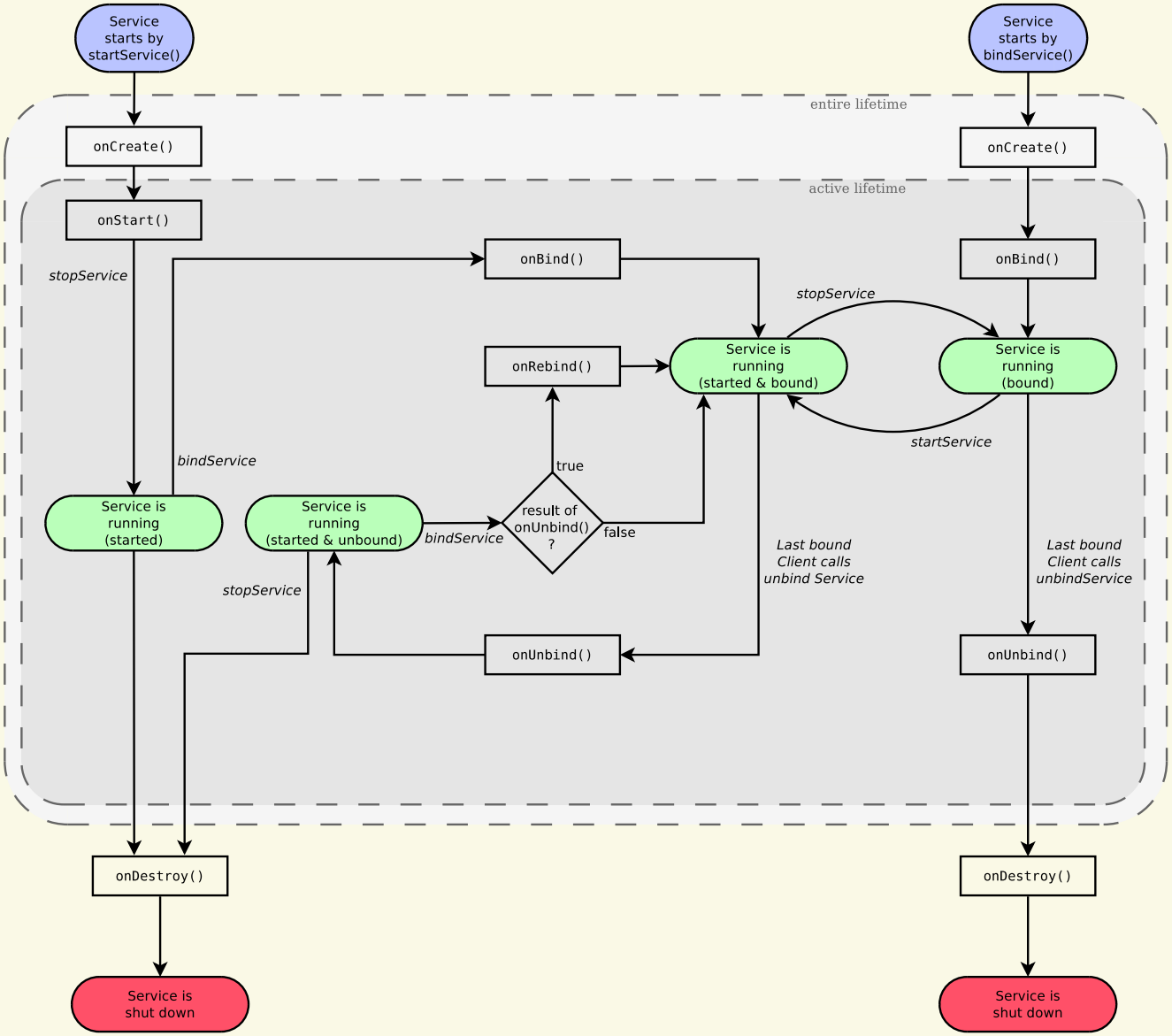
Android-Framework bietet **Livecycle-Methoden** Grund:

- Applications auf einem Telefon sollten unterbrechbar sein
- Modularität ermöglicht Wiederverwendung (lose Kopplung)

Activity


Activity starts
entire lifetime
User navigates back to the activity
onCreate()
visible lifetime
onStart()
onRestart()
Process is killed
foreground lifetime
onResume()
Activity is running
Another activity comes in front of the activity (or *)
onPause()
Other applications need memory
Activity is paused
The activity comes to the foreground
The activity is no longer visible (or * and **)
onStop()
Activity is stopped
The activity comes to the foreground
The activity is or will be finishing by himself via finish() (*) destroyed by the system (**)
onDestroy()
Activity is shut down

Service


Service starts by startService()
Service starts by bindService()
entire lifetime
onCreate()
onCreate()
active lifetime
onStart()
onBind()
onBind()
stopService
stopService
onRebind()
Service is running (started & bound)
Service is running (bound)
startService
bindService
true
Service is running (started)
Service is running (started & unbound)
result of onUnbind() ?
bindService
false
Last bound Client calls unbind Service
Last bound Client calls unbindService
stopService
onUnbind()
onUnbind()
onDestroy()
onDestroy()
Service is shut down
Service is shut down