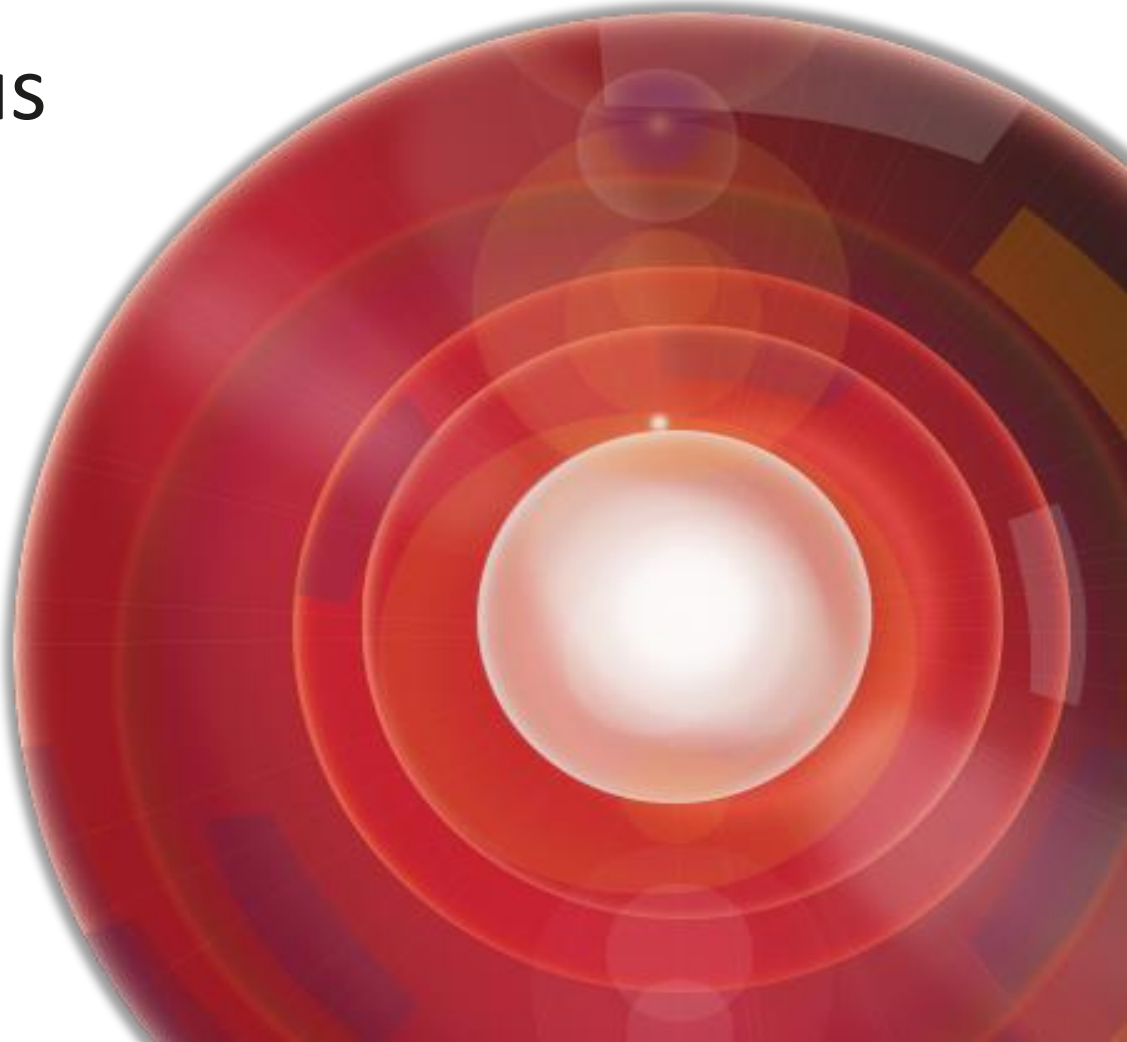
comlet
Engineering your success

Security Tests im Autopilot-Modus

Herausforderungen und Möglichkeiten automatisierbarer Penetration Tests



Michael Eisenbarth, comlet Verteilte Systeme

ESE-Kongress 2017, 07.12.2017, Sindelfingen

Innovation wächst aus der Kombination von Erfahrung und Talent

2001 gegründet von Professoren der Hochschule Kaiserslautern, Standort Zweibrücken

Firmensitz Zweibrücken, Campus der Hochschule Kaiserslautern

Niederlassungen Stuttgart, Darmstadt, München

Workforce Rund 80 Experten im Bereich Embedded Systems & starken Partnern, Eigener Freelancer und Expertenpool

Michael Eisenbarth
Leiter Fachbereich
Distributed Security

Jasmin Koch
Leiter Fachbereich
Testing & Validation

Steffen Fromm
Geschäftsführer

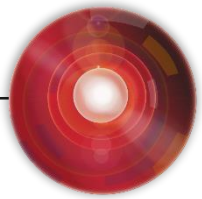
Achim Mayer
Leiter Fachbereich
Configuration Management

Björn Decker
Leiter Fachbereich
System Architecture

comlet

Begleitung über den gesamten Entwicklungszyklus Ihres Produktes

comlet



Unsere Fachbereiche für Ihre Spezialthemen

Configuration Management

- Unterstützung bei der Handhabung von Produktvarianten
- Software- und Systemintegration
- Automatisierte Buildprozesse
- Continuous Integration
- Releasemanagement

Testing & Validation

- Statische Codeanalysen
- Unittests
- Integrationstests
- Softwaretests
- Systemtests
- Testautomatisierung und TDD

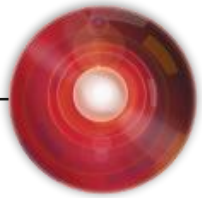
Distributed Security

- Standard Compliance Überprüfungen
- Bedrohungs- und Risikoanalysen
- Sichere Softwareentwicklung und Architekturdesigns
- Securitytests

System Architecture

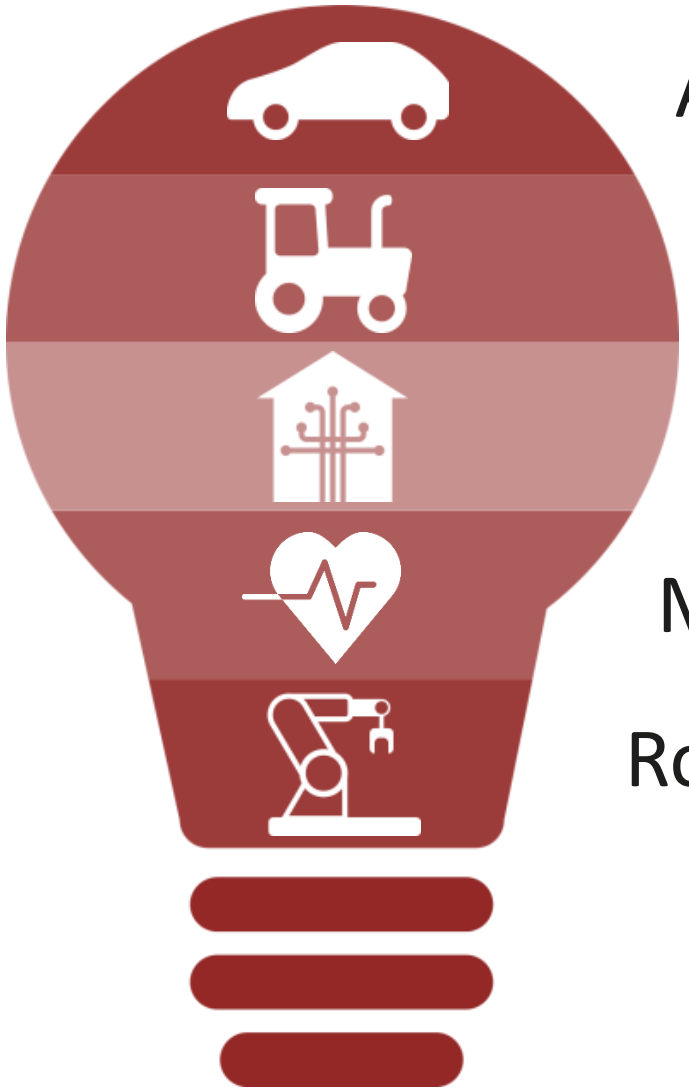
- Architektur-Analyse/Beratung
- Framework-Entwicklung
- Codereviews
- Re-Engineering und Re-Strukturierung
- Systemanalyse und Performance-optimierungen

comlet

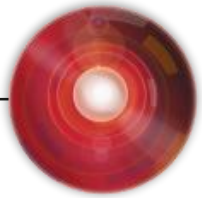


- Motivation
- Hintergrund Penetrationtesting
- Speziallfall Embedded-Systems
- Case-Study
  - Aufbau
  - Automatisierte Angriffsvektoren
  - Fazit
- Lessons Learned

Embedded Software für Ihre Produkte



- Automotive
- Agriculture & Forestry
- Smart Home & Living
- Medical Devices
- Robotics & Industry



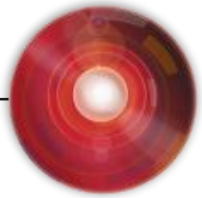
Technologie

- Legacy-Architekturen sind noch nicht für Vernetzung ausgelegt
- Anzahl der Steuergeräte, Sensoren und Schnittstellen wächst
- Steuergeräte in verschiedenen Systemdomänen sind über Backbones verbunden
- Entwicklung hin zu Ethernet-Netzwerken, die TCP/IP und andere bekannte Protokolle verwenden
- Vielzahl externer Verbindungen wie LTE, V2X, WiFi, BT, NFC, Ladestation, USB, …
- Firmware muss schnell / häufig aktualisiert werden

Markt

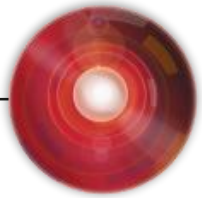
- Boom im Bereich Vernetzung
  - Verbreitung von Embedded-Systems nimmt zu
  - Alles wird vernetzt und „smart“
- Meist unter Preisdruck entwickelte Geräte
  - Sicherheit nicht im Fokus
- Kriminelle professionalisieren sich
  - Automatisierte Infektionen im großen Stil, Botnetze
  - Dienstleistungsangebote (Exploit-Kits) im Darknet auch von „Laien“ nutzbar

comlet



Besonderheiten von Security-Anforderungen

- Security ist eine nicht-lokale, globale Systemeigenschaft
  - Kein offensichtlicher oder direkter Zusammenhang zwischen abstraktem Security-Ziel, z.B. Vertraulichkeit und technischer Umsetzung der erforderlichen Schutzmaßnahme erkennbar
- Security ist relativ (zum Bedrohungsmodell)
  - Absolute Sicherheit meist nicht realisierbar (oder nicht bezahlbar)
- Security nicht invariant gegenüber Verfeinerung oder Komposition
  - Erschwert schrittweise Verfeinerung oder Divide&Conquer-Strategien
- Erfolgreiche Umsetzung (selbst im Erfolgsfall) schwer messbar
  - fehlende Metriken für Security (anders als z.B. bei Performance)

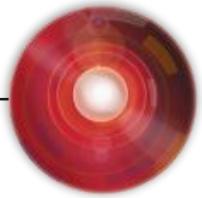


Exkurs: Messbarkeit (Testbarkeit) von Security-Anforderungen

- **Lord Kelvin (1846-99, Prof. theoretische Physik):**
  »I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the state of science, whatever the matter may be.«

- **Tom DeMarco (1982 [widerrufen 2008/2009]):**
  »You cannot control what you can’t measure.«

- **Albert Einstein:**
  »Not everything that counts can be counted, and not everything that can be counted counts«.

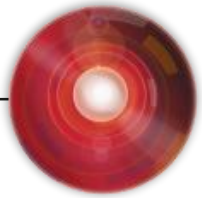


Angriff als die beste Verteidigung?!

- Angriff:
  - Von sogenannten „Blackhat“-Hackern durchgeführt
  - Hat bösartige Absicht, hält Sicherheitslücken geheim
  - Ziele, u.a.: Profit, Manipulation, Vandalismus, Spionage,…

- Verteidigung / Penetrationtesting:
  - Von „Grey-“ bzw. „Whitehat“-Hackern durchgeführt
  - Sucht Schwachstellen um diese zu schließen / bekannt zu machen
    - Greyhat: Tut dies ohne Einwilligung und Vorwarnung des Herstellers
  - Ziele: Verbesserung der Sicherheit und Transparenz

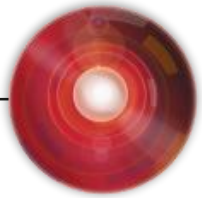
comlet



Arten von Penetrationstests

- Blackbox:
  - Keine Informationen verfügbar
  - Breitflächige Tests notwendig → auch zur Informationsgewinnung
- Whitebox:
  - Entwicklungs-Informationen verfügbar (Doku, Code, Interviews, Zugangsdaten)
  - Tiefergehende Tests und Analysen möglich
- Greybox:
  - Kombination aus Black- und Whitebox, idealerweise durch unterschiedliche Teams
  - Zu Beginn wenige Informationen bekannt
  - Breitflächige und tiefgehende Tests möglich

→ Greybox vereint Vorteile von Black- und Whitebox-Penetrationstests



Herausforderung Embedded-Systems

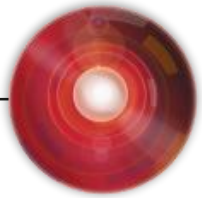
- Penetrationtesting-Werkzeuge und Anleitungen meist noch auf PC und Netzwerktechnologie fokussiert
  - (WLAN, Port-Scans, Protokolle, SSL, Phishing etc.)
- Embedded-Systems sind meist sehr individuell
  - Vielzahl von Schnittstellen
  - Vielfältige Hardware (Prozessor-Architektur)
  - Wenig „Standard-Software“
  - Dauerbetrieb
  - Ggf. wenig bis keine Benutzerinteraktionen

→ Angreifer wird sich explizit mit dem Gerät befass

→ Beschafft sich Zugriff auf die Hardware

„Attackers Dilemma“

comlet

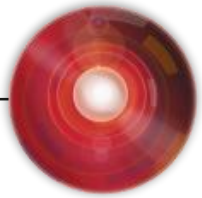


Eine paar Möglichkeiten, um Sicherheitsprobleme zu finden…

- Statische Code-Analyse, z.B. Analyse nach Common Weaknesses (CWE – Mitre)
- Fuzzing oder andere dynamische Tests
- Penetration Tests
- Red Teams
- Auf die Veröffentlichung von Fehlerberichten warten ;-)
- …

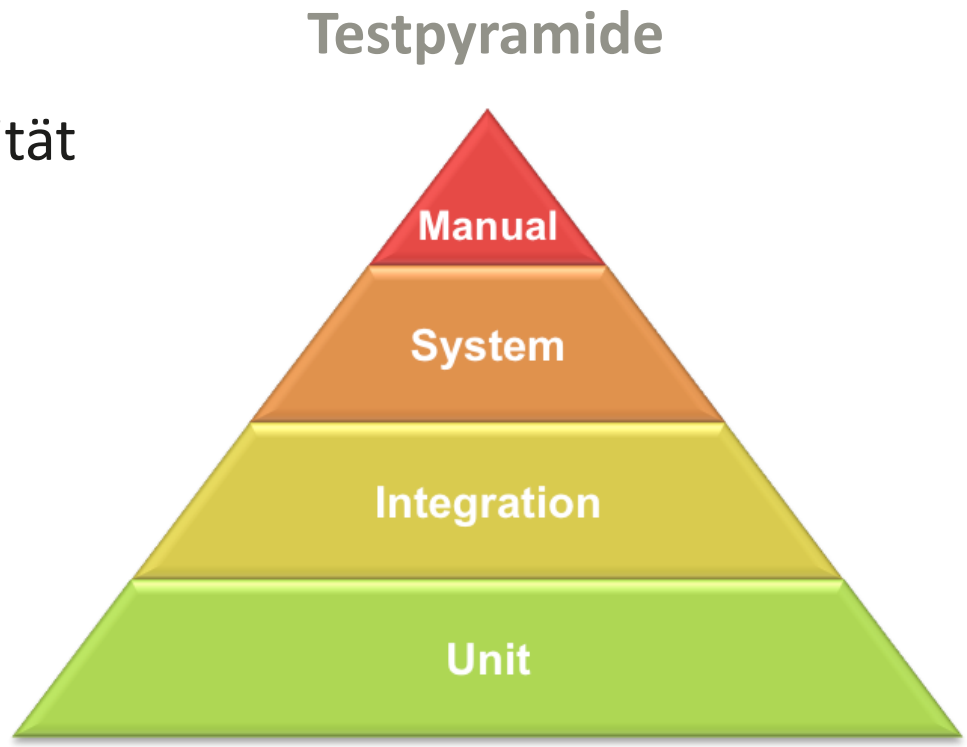
→ Hierbei werden Probleme natürlich erst sehr spät im Pr rkannt!

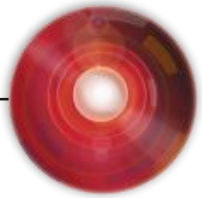
„Defenders Dilemma“



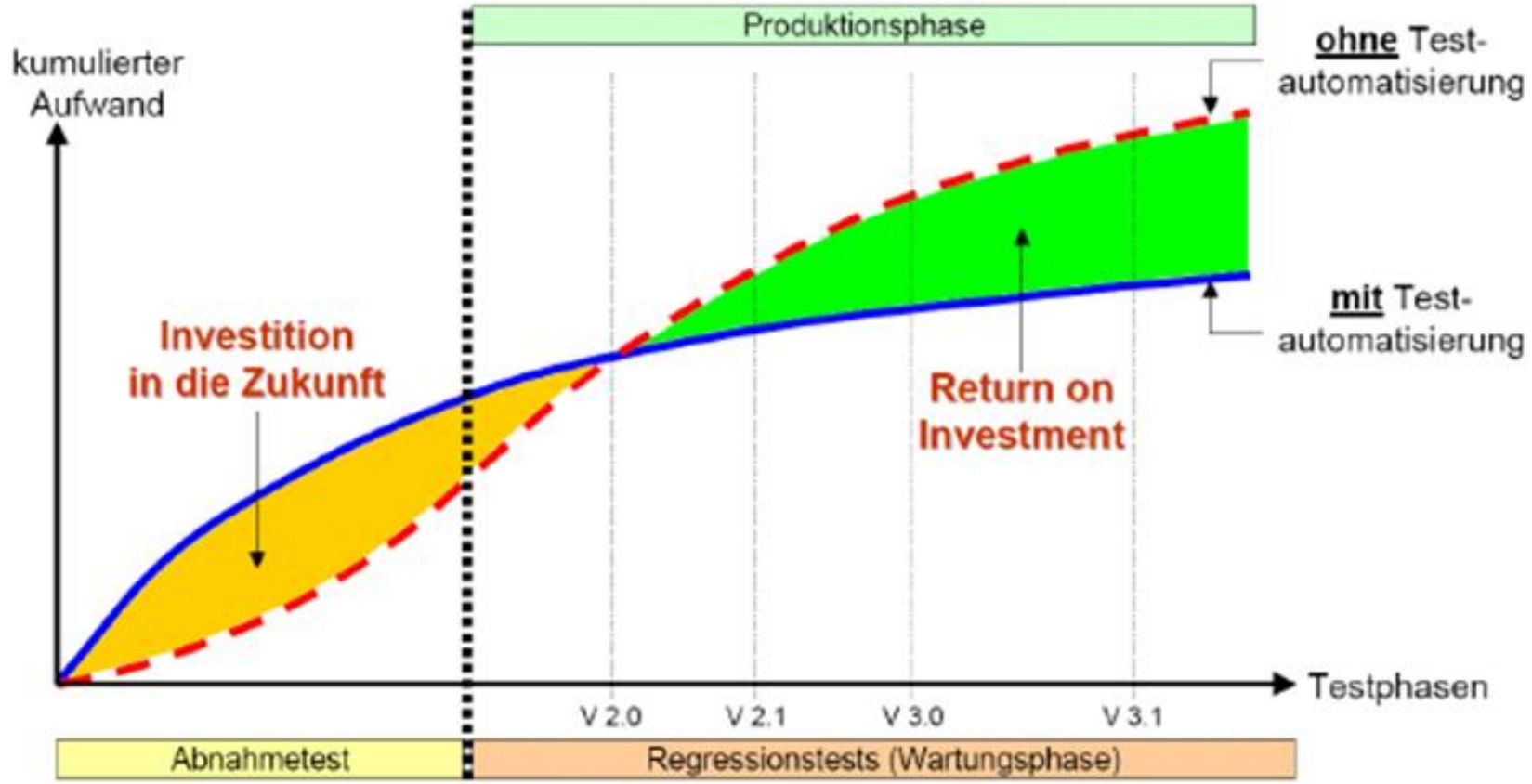
Vorteile Testautomatisierung

- Schnellere Rückmeldung – „Nightly Builds“
- Höhere Testabdeckung
- Reproduzierbarkeit, Erhöhung der Testqualität
- Zeitersparnis System Test
- Höhere Softwarequalität
- …

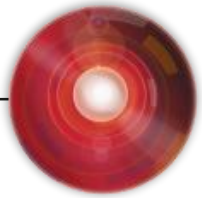




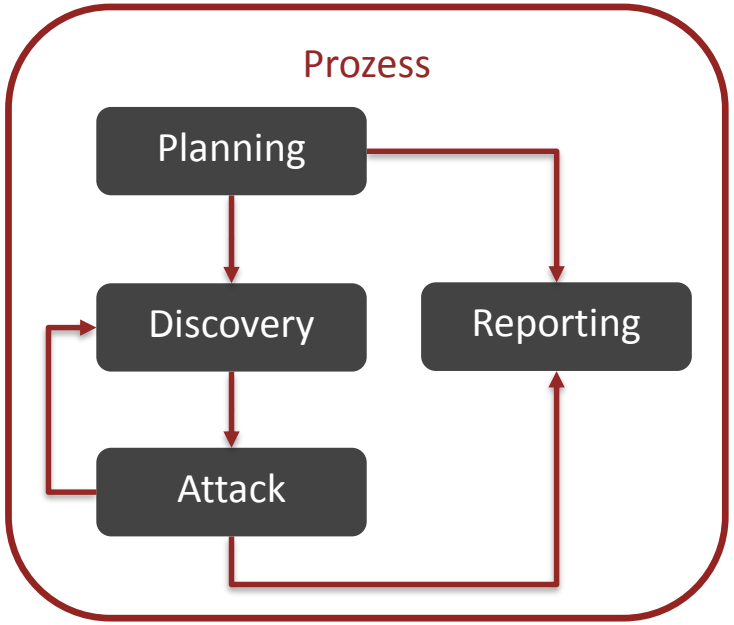
Testautomatisierung als Investition für die Zukunft



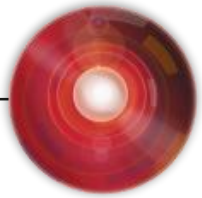
Quelle: Wikipedia



Security Tests: Phasen & Arten



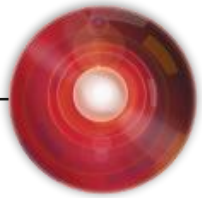
- modellbasierte Sicherheit
  - Tests basieren auf Anforderungen und Designmodellen, die während der Analyse- und Designphase erstellt wurden,
- Code-basiertes Testen und statische Analyse
  - auf Source-und Byte-Code während der Entwicklung erstellt,
- Penetrationtests und dynamische Analysen
  - auf laufenden Systemen, entweder in einer Test- oder Produktionsumgebung,
- Sicherheits-Regressionstest
  - während der Wartung durchgeführt



Ausgangspunkt bildet eine Risiko / Bedrohungsanalyse

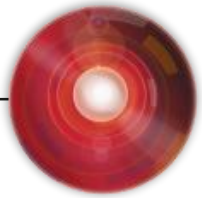
- Das Ziel der Risiko- / Schwachstellenanalyse besteht darin,
  - spezifische Schwachstellen und Bedrohungen für die Software zu ermitteln
  - und deren Auswirkungen zu bewerten.

- Gegenstand:
  - Mögliche Bedrohungen (Threats) in jeder Ebene (oder Komponenten)
  - Arten von Sicherheitslücken, die in den Komponenten vorhanden sein können
  - Unternehmerische Auswirkungen (Folgen und Kosten eines erfolgreichen Angriffs)
  - Eintrittswahrscheinlichkeit
  - bestehende und empfohlene Gegenmaßnahmen zur Minderung identifizierter Risiken



Ziel 1: Überprüfung der aktuellen Systemhärtung (Schutz vor unerlaubtem Eindringen)

- Sub-Ziele:
  - Schließen (ggf. in Produktivumgebung) ungenutzter Netzwerk-Ports nach außen
  - Deaktivierung von ungenutzten USB-Ports zur Kontrolle verwendeter Peripheriegeräte
  - Keine überflüssigen Schnittstellen: Vor allem Entwickler- und Debug-Schnittstellen sollten in der Release-Version entfernt werden
  - Nicht unterbrechbarer Bootloader
  - Aktive Schutzmechanismen gegen Stack-Manipulation

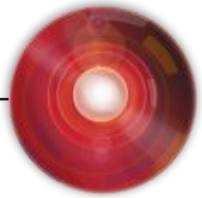


Ziel 2: Blackbox-Fähigkeiten (Analyse von außen erschweren)

- Sub-Ziele:
  - Verwendung aktueller kryptographischer Verfahren
  - Verwendung von TLS für Netzwerkverkehr mit Überprüfung der Serverzertifikate
  - Verwendung von asymmetrische Verschlüsselung von Software-Updates

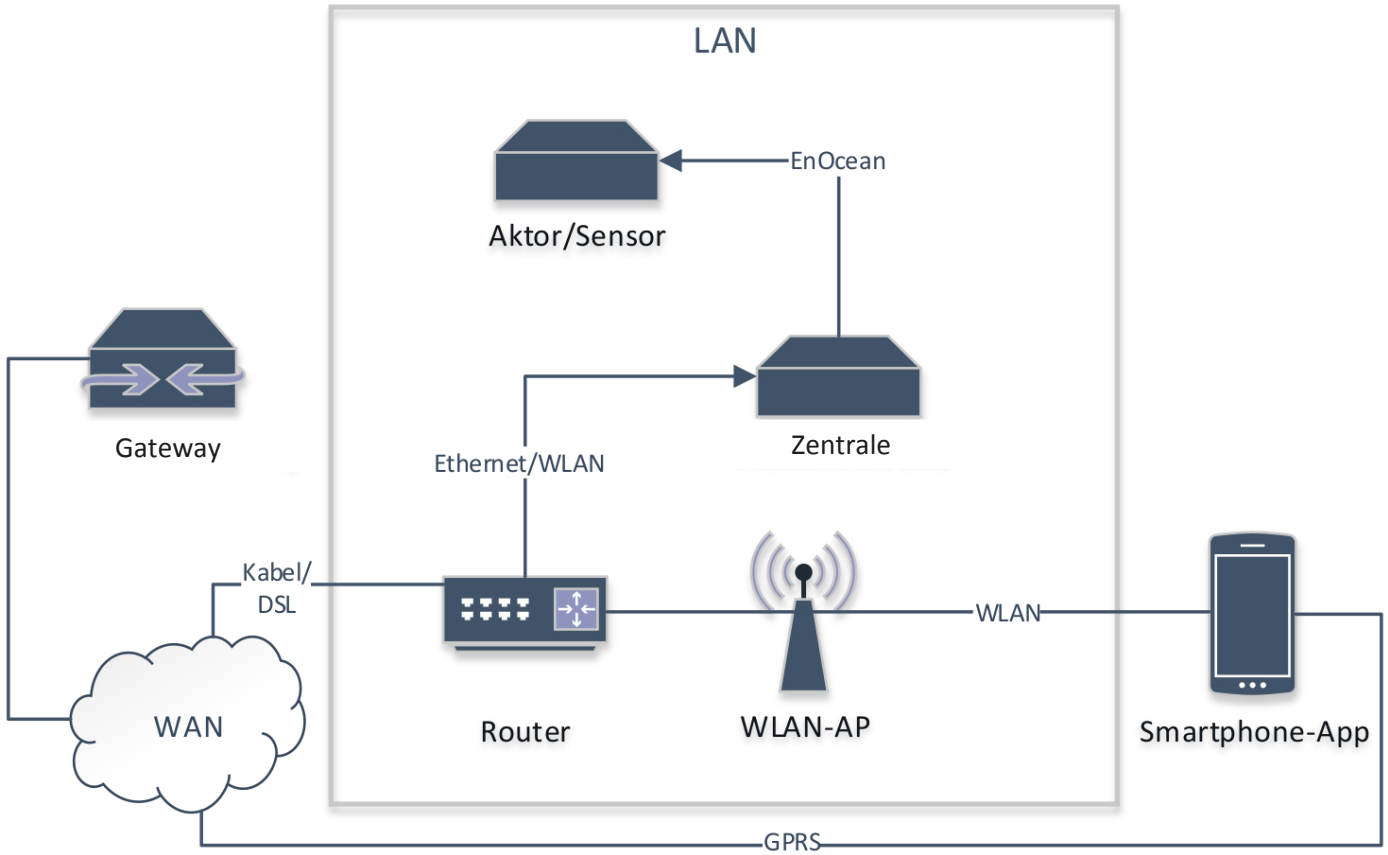
Ziel 3: Möglichkeiten der Post Exploitation (Analyse nach dem Eindringen erschweren)

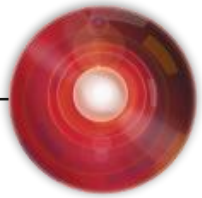
- Sub-Ziele:
  - Deaktivierung von Prozessverfolgung im Kernel zur Erschwerung von Debugging und Reverse Engineering
  - Kernel-Konfiguration nicht im laufenden System einsehbar
  - Auslieferung eines möglichst reduzierten Systems
  - Minimierung der Log-Ausgaben in der Auslieferungsversion
  - Kompilierung der Auslieferungsversion ohne Symboltabelle



Case-Study

- Sicherheitsanalyse eines Heimautomatisierungssystem
  - Aktor/Sensor Steuerung über EnOcean-Funkprotokoll
  - Verbindung „Gateway“ (Internet-Server)
  - Steuerung mit Smartphone-App (LAN direkt, Mobil über Gateway)
  - Linux System auf ARM-Architektur





Case-Study – Auszug automatisierbare Testfälle

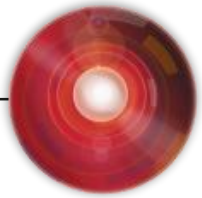
Testfallname: DSEC_1_PORTSCAN

Komponente/Funktion: Netzwerk
Verknüpfung zu Anforderungen: <Eindeutiger Verweis auf die Anforderungen kann z.B. Kapitelnummer oder AnforderungsID in der Kundenspezifikation sein >

Voraussetzungen: Das Testobjekt verfügt über eine aktive Ethernet oder WLAN Schnittstelle. Aktiv heißt, dass eine Verbindung zum LAN mit gültiger IP-Adresse besteht. Das Testobjekt befindet sich im regulären Betriebsmodus.

Beschreibung: *Mit Hilfe eines Port Scanners (z.B. nmap) wird sichergestellt, dass nur die in der Spezifikation notwendigen UDP und TCP Ports nach außen (zum LAN) geöffnet sind. Ports die z.B. für die Entwicklung notwendig sind, sollten in der Release-Version nicht mehr verfügbar sein.*
Der Port Scan kann mit dem Befehl `nmap -p 1-65535 -T4 -A -v -sS -sU <IP>` ausgeführt werden.

Erwarteten Ergebnisse: Der Port Scan liefert nur die notwendigen Ports mit Status „open“ zurück.



Case-Study – Auszug automatisierbare Testfälle

Testfallname: DSEC_5_STACK_PROTECTION

Komponente/Funktion: Software
Verknüpfung zu Anforderungen: <Eindeutiger Verweis auf die Anforderungen kann z.B. Kapitelnummer oder AnforderungsID in der Kundenspezifikation sein >

Voraussetzungen: Das Testobjekt verfügt über ein Linux Betriebssystem das die Stack- und Buffer-Overflow-Protection Mechanismen ASLR und EDB (NX) unterstützt. Der Tester hat Zugang zur Konsole des Linux Systems des Testobjekts.

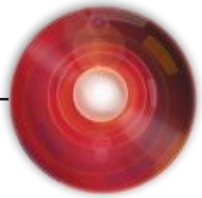
Beschreibung: Der Test überprüft, ob

- *die Programme des Testobjekts als Position Independent Code (PIC) kompiliert wurden,*
- *die Programme mit Stack Protector kompiliert wurden,*
- *das Betriebssystem des Testobjekts die Address Space Layout Randomisation (ASLR) aktiviert hat und*
- *der Kernel des Testobjekts mit aktiviertem Execute Disable Bit (EDB, NX) ausgeführt wird.*

Erwarteten Ergebnisse:

- ASLR ist aktiviert, wenn in /etc/sysctl.conf **kernel.randomize_va_space** auf **2** gesetzt ist.
- Execute Disable ist aktiviert, wenn in /proc/cmdline **noexec=on noexec32=on** gesetzt ist.
- Position Independent Code ist aktiviert wenn, **objdump –R** RELATIVE Relocations anzeigt.

comlet



Case-Study – Auszug automatisierbare Testfälle

Testfallname: DSEC_10_ STRIP_SYMBOL_TABLE

Komponente/Funktion: Anwendung
Verknüpfung zu Anforderungen: <Eindeutiger Verweis auf die Anforderungen kann z.B. Kapitelnummer oder AnforderungsID in der Kundenspezifikation sein >

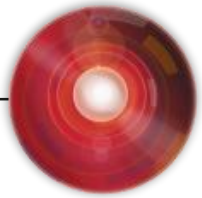
Voraussetzungen: Der Tester hat Zugang zu den Executables oder der Build-Umgebung der Programme des Testobjekts.

Beschreibung: Mit dem Test wird sichergestellt, dass die Executables in der Release-Version keine Symbol-Tabellen enthalten.

Dadurch sind die Namen von Methoden, Klassen, Funktionen und Variablen, wie sie im Quellcode vorliegen, nicht mehr vorhanden. Dies erschwert einem Angreifer die Analyse.

- Mit dem Befehl objdump –t <EXECUTABLE> kann die Existenz der Symbol-Tabelle überprüft werden.

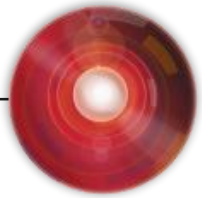
Erwarteten Ergebnisse: objdump erzeugt die Ausgabe
- SYMBOL TABLE: no symbols



Case-Study Fazit – potentielle Risiken

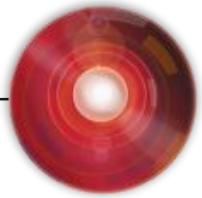
- unsicherer Boot-Prozess
  - Bootloader über serielle Schnittstelle unterbrechbar
- Software-Update-Schwachstellen
  - Update-Paket wird zyklisch per HTTP abgefragt
    - URL + POST-Parameter mit tcpdump sichtbar
  - Archiv ist unverschlüsselt, enthaltene Dateien einzeln verschlüsselt + signiert
    - Eine korrekt verschlüsselte und signierte *.enc Datei im Archiv genügt
- Volle Kontrolle über die Zentraleinheit (per root-Zugang)
  - Pre- und Post-Install Bash-Skripte werden mit root-Rechten ausgeführt
  - Beliebige Dateien können so im System installiert werden, z.b. SSH Daemon, reverse Shell, Bot, …
- Im LAN konnte Software-Paket mittels„Joonior“-Protokoll auf ZE gepusht werden!
  - z.B. simulierte Smartphone-App, infiziertes Smartphone
  - Kein Zutun oder Bestätigung des Benutzers notwendig
  - Automatische Installation im Hintergrund

comlet



Case-Study Fazit – abgeleitete Gegenmaßnahmen

- Bootloader härten
  - Unterbrechen nur mit vorheriger Authentifizierung
- Software-Update Prozess verbessern
  - Archiv verschlüsseln + signieren, anstelle oder zusätzlich zu einzelnen Dateien
  - Update-Paket nur über HTTPS, mit Client-Authentifizierung
  - „Joonior“-Protokoll absichern
- Dateisystem vor Manipulation mit Secure-Boot Mechanismus schützen
  - RO-RootFS (z.B. squashfs oder ramdisk) + Signatur aus Bootloader prüfen
  - Keine ausführbaren Dateien in RW-Partition
- Fehler im Build-System beheben (Debug/Release)



Laboraufbau und Lessons Learned

- Tester muss kreativ sein und improvisieren können
  - Erfordert Kompetenz und Erfahrung!
  - breites Spektrum an Programmierkenntnissen
  - Hardware und Elektro-Kenntnisse von Vorteil
- Je nach zu testendem Produkt andere Tools notwendig
  - Ggf. müssen neue Tools individuell entwickelt werden
- Viele Tools machen nur in Kombination Sinn
- Schnellere Rückmeldung hinsichtlich Security Schwachstellen durch „Nightly Builds“ ist möglich
- Integration von Common Weakness Enumeration in statische Code Analyse Werkzeuge
- Reproduzierbarkeit und Erhöhung der Testqualität erreicht
- „Bewusstsein der Entwickler und Tester hinsichtlich Security wird geschärft!

comlet