softScheck
we identify vulnerabilities others don't

Consulting Angebot IT-Sicherheit

Darstellung der Verfahren Security Design, Threat Modeling, Static Source Code Analysis und Dynamic Analysis: Fuzzing, Explorative Testing, Konformitätsprüfung

Prof. Dr. Hartmut Pohl
Geschäftsführender Gesellschafter

softScheck GmbH

Stand: April 2013

Inhaltsverzeichnis

Einleitung

Wir beraten im Bereich Informationssicherheit / IT-Sicherheit seit über 10 Jahren große, mittlere und kleine nationale und internationale Unternehmen und Behörden mit heute insgesamt 12 Mitarbeitern, die unter anderem auf folgende Bereiche spezialisiert sind:

- Threat Modeling, Static Analysis und Dynamic Analysis: Fuzzing zur kostengünstigen Identifizierung bislang nicht-erkannter Sicherheitslücken
- Penetration Testing: Erkennen bekannter Sicherheitslücken in Software wie Standardsoftware und Individualentwicklungen
- Gerichtsfeste forensische Untersuchungen doloser Handlungen
- Business Continuity und Desaster Recovery

Wir arbeiten in allen diesen Bereichen so weit wie möglich Tool-gestützt und setzen die weltweit wirkungsvollsten Tools ein bzw. nutzen ggf. auch Eigenentwicklungen.

Darüber hinaus sind wir erfahren in den Bereichen:

- Coaching von Sicherheitsbeauftragten
- Beratung von Beratern in der Identifizierung neuer Sicherheits-Arbeitsfelder und Beratung von Herstellern bei der Planung und Entwicklung innovativer Sicherheitsprodukte

softScheck bietet seit Jahren als Alleinstellungsmerkmal in Europa erfolgreich die kostengünstige Identifizierung bisher nicht-erkannter (!) Sicherheitslücken (Zero-Day-Vulnerabilities) in kommerzieller Software, in Industrial Control Systems (ICS), SCADA, im Smart Grid / Smart Meter Gateway (SMGW) und auch in Hardware an und übernimmt für Sie auch den gesamten Security-Testing Process.

Darüber hinaus bieten wir selbstverständlich auch die klassische IT-Sicherheitsberatung an - vom Grundschutz (ISO 27000-Familie inkl. Governance, Risk Analysis und Compliance) bis hin zur Hochsicherheit in der Informationsverarbeitung (Redundanz und Diversität).

In diesem Leistungskatalog finden Sie Details zu allen unseren Beratungsfeldern.

Ihre IT-Systeme leisten damit Angriffen von Innen- und Außen nachhaltigen Widerstand. Gern beraten wir Sie – unter Einhaltung der gewünschten Diskretion im Rahmen eines persönlichen Gesprächs.

Prof. Dr. Hartmut Pohl
Geschäftsführender Gesellschafter
softScheck GmbH

Management Summary

Angesichts der beiden Fakten:

- Software kann nicht fehlerfrei erstellt werden; macht dies eine Überprüfung der Software erforderlich und
- kein (erfolgreicher) Angriff ohne Ausnutzung einer Sicherheitslücke

bieten wir "Security Testing as a Service" als vollständigen Prozess an, um das Sicherheitsniveau der Software unserer Kunden zu steigern. Zur Identifizierung bisher nicht-erkannter Sicherheitslücken arbeiten wir dazu erfolgreich Tool-gestützt mit den – auch vom Bundesamt für Informationssicherheit (BSI) unterstützten – folgenden 7 Verfahren:

1. **Security by Design**: Entwicklung von Sicherheitsarchitekturen für Software
2. **Threat Modeling**: Überprüfung der Sicherheitsarchitektur
3. **Dynamic Analysis – Fuzzing**: Test der ausführbaren, kompilierten Datei – kein Quellcode nötig. Wir setzen im Einzelfall über 50 Tools aus den weltweit mehr als 300 verfügbaren ein: Jedes Fuzzing-Tool identifiziert andere Sicherheitslücken!
4. **Static Source Code Analysis**: Überprüfung von Implementierungsfehlern
5. **Penetration Testing**: Zur Überprüfung auf bereits bekannte Sicherheitslücken
6. **Explorative Testing** und manuelles Code Auditing.
7. Und letztlich identifizieren und analysieren wir auch **Covert Functions** – undokumentierte, verdeckte Funktionen – u.a. auch auf Smartphones und mobile Devices

Im Folgenden sollen insbesondere die 3 erfolgreichen Verfahren Threat Modeling (Design Phase), Static Source Code Analysis (Implementation Phase) und Dynamic Analysis: Fuzzing (Verifikation Phase) dargestellt werden; mit diesen Verfahren werden insbesondere alle bisher nicht-veröffentlichten Sicherheitslücken (Zero-Day-Vulnerabilities) identifiziert. Der Einsatz dieser 3 Verfahren wird umso wirkungsvoller und kostengünstiger je früher in der Software-Entwicklung Sicherheitslücken identifiziert werden möglichst also beginnend in der Designphase. Am teuersten wird es, wenn die Software schon ausgeliefert ist (Release Phase) und erst dann korrigiert wird - vgl. Abb. 1.

Zudem lassen sich diese 3 Verfahren für viele Anwendungsbereiche einsetzen: Anwendungssoftware (Webapplications, ERM, CRM, SCM, ERP, E-Business, CIM etc.) und Netzwerk-Protokolle, Embedded Systems (auch die Hardware) und Industrial Control Systems (ICS) (auch proprietärer Systeme) Manufacturing Execution Systems (MES), Produktionsleitsysteme, SCADA (Leittechnik und –systeme), SPS bis zur Feldebene, Cyber Physical Systems (CPS), Industrie 4.0, in Apps und Applets für smart and mobile Devices, im Cloud Computing und auch in Hardware. Im Bereich Smart Grid / M2M haben wir erfolgreich Energy Management Systeme - EMS und Machine-to-Machine Communication in einem Smart Meter Gateway (SMGW) abgesichert.

Der Einsatz herkömmlicher Verfahren zur Behebung von Fehlern und insbesondere bisher nicht erkannter Sicherheitslücken ist dagegen sehr kostenaufwändig - viele Sicherheitslücken werden daher nicht oder erst nach der Auslieferung der Software an die Kunden - z.T. auch von Dritten - erkannt.

Aus eigenen Untersuchungen wissen wir, dass der Aufwand zur Identifizierung von Sicherheitslücken mit Threat Modeling, Static Source Code Analysis und Dynamic Analysis: Fuzzing vergleichsweise sehr kostengünstig durchgeführt werden kann – auch wenn sich die Software schon auf dem Markt befindet.

Neben der Identifizierung bisher nicht-erkannter Sicherheitslücken werden diese bewertet (Rating), um entscheiden zu können, ob alle identifizierten Sicherheitslücken behoben werden sollen oder ob auf das Patchen einiger verzichtet werden kann (Priorisierung).

Die entscheidenden Faktoren für diese drei Verfahren sind:

- Wir empfehlen den Einsatz der drei Verfahren Threat Modeling, Static Source Code Analysis und Dynamic Analysis: Fuzzing, weil sie sich erfolgreich ergänzen.
- Dynamic Analysis: Fuzzing benötigt zum Erfolg keinen Quellcode (Blackbox-Test): Die Software wird während Ihrer Ausführung untersucht. Hierzu kann die Software in einer Virtuellen Maschine oder auf einem anderen Testsystem ausgeführt werden.
- Wir empfehlen den Einsatz von bis zu 50 Fuzzing-Tools, weil nur ein derart großes Bündel von Fuzzern eine hinreichende Menge kritischer Sicherheitslücken identifiziert.

Ist für ein konkretes Target System ein angemessener Fuzzer nicht verfügbar, entwickelt softScheck spezielle Fuzzer. Für spezielle Target Systems entwickelt softScheck auch spezifische Attack Strings auf der Basis selbstentwickelter Testdaten.

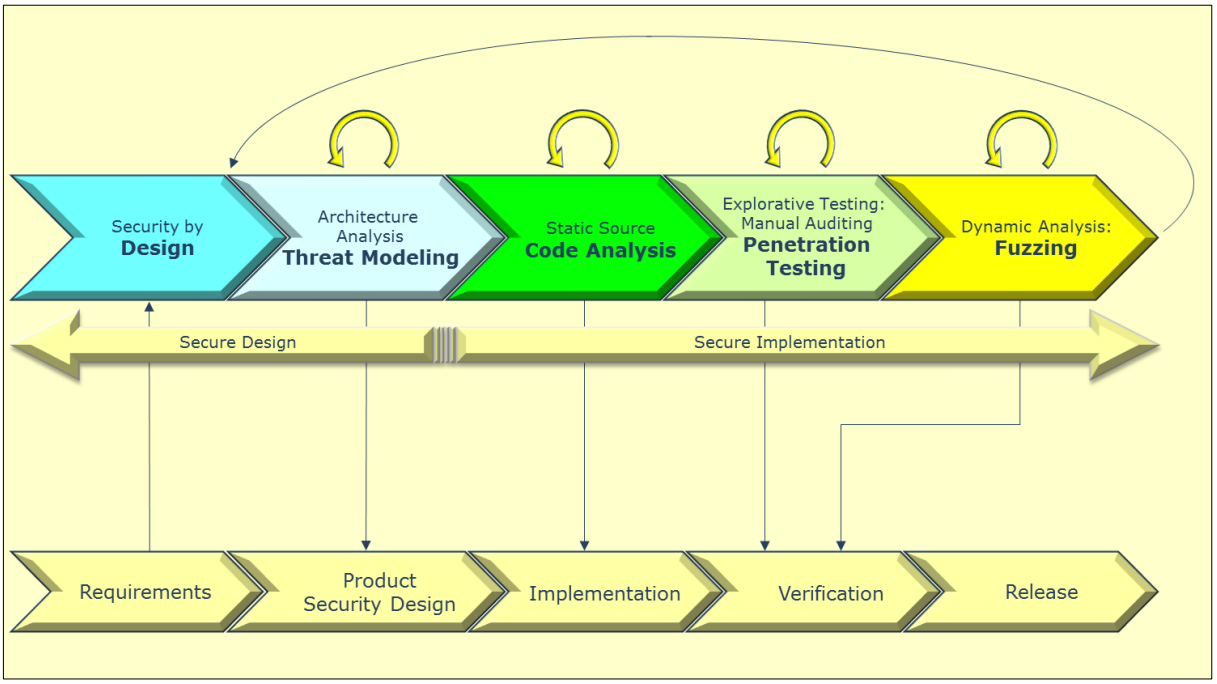

Abb. 1: softScheck Security Testing Process

Mit der Identifizierung und Behebung bisher nicht-erkannter Sicherheitslücken kann sporadischen Betriebsausfällen und unbeabsichtigten Datenabflüssen - den häufigsten Folgen sicherheitsrelevanter Softwarefehler - entgegengewirkt werden und so können proaktiv hohe Umsatzausfälle, Datenschutzprobleme und Reputationsschäden gemindert werden.

Mit den von uns sowohl Tool-gestützt als auch mit Manual Auditing eingesetzten Methoden sind wir sehr erfolgreich. In einer Stichprobe aus 40 Projekten haben wir durchschnittlich insgesamt 156 Sicherheitslücken identifiziert:

Method	No. Identified Zero-Day-Vulnerabilities	No. Tools used
Architecture Analysis: **Threat Modeling**	**112 (986)**	1
Static Source Code Analysis	17	3 - 5
Penetration Testing	0 (76)	4 +
Dynamic Analysis: **Fuzzing**	**27**	> 60
Summe	**156**	**> 70**

Das Bundesamt für Sicherheit in der Informationstechnik empfiehlt den Einsatz von Threat Modeling, Static Source Code Analysis und Dynamic Analysis: Fuzzing.

softScheck **hat in den geprüften Zielprogrammen mit dem hier dargestellten Security Testing Process tatsächlich alle bis dahin nicht-bekannten Sicherheitslücken identifiziert. Nach diesen Sicherheitsprüfungen wurden also keine anderen / weiteren Sicherheitslücken mehr erkannt!**

1 Consulting Security Design

Das Consulting zum Security Design erfolgt in der Designphase im Hinblick auf die Übereinstimmung der Security Aspekte des Produktdesigns mit den Anforderungen – z.B. der Technischen Richtlinie TR 03109 für ein Smart Meter Gateway (SMGW) sowie dem Protection Profile for the Gateway of a Smart Metering System (Smart Meter Gateway PP). Zusätzlich wird untersucht, welche weiteren Maßnahmen zur Gewährleistung der Sicherheit eines Produkts vom Auftraggeber gefordert werden.

Vorgehen bei der Prüfung:

- Sichtung der Security Design-Dokumentation
- Auflistung identifizierter Sicherheitsmängel, die fehlende Übereinstimmungen mit der technischen Richtlinie enthält sowie Mängel in den weiteren Sicherheitsmaßnahmen. Die Ergebnisse werden ergänzt durch vorgeschlagene Behebungsmaßnahmen.
- Dokumentation umgesetzter und nicht umgesetzter Anforderungen
- Erstellung des Abschlussberichts
- Nach Behebung der im Abschlussbericht dokumentierten Mängel durch den Auftraggeber führt softScheck in jedem Fall eine Wiederholungsprüfung durch
- Das Projekt wird mit einer Präsentation der Ergebnisse im Hause des Kunden abgeschlossen.

Sichtung der Security Design Dokumentation

Die vom Auftraggeber zur Verfügung gestellten Dokumente zum Security Design werden gesichtet, systematisch strukturiert und für eine Analyse aufbereitet.

Bewertung des Security Design Dokumentation

Jede in der Security Design Dokumentation beschriebene sicherheitsrelevante Funktion wird auf das Einhalten der Anforderungen überprüft.

Auswertung: Validierung aufgetretener Mängel

Jeder festgestellte Mangel wird einer Zweitprüfung unterzogen, um False Positives auszuschließen.

Hinweise zu Behebungsmaßnahmen

Es werden Behebungs- und/oder Umgehungsmaßnahmen vorgeschlagen.

Abschlussbericht

Der Abschlussbericht beinhaltet die identifizierten Mängel sowie Vorschläge zu entsprechenden Behebungsstrategien und nimmt eine Bewertung des Sicherheitsniveaus des Security Designs vor. Es werden alle für den Auftraggeber relevanten Daten in diesem Dokument zusammengefasst dokumentiert.

Wiederholungsprüfung

Nach den Sicherheitskorrekturen durch den Auftraggeber wird das Security Design insbesondere unter Berücksichtigung dieser vorgenommenen Korrekturen erneut geprüft.

Präsentation der Ergebnisse

Präsentiert werden alle Ergebnisse von Prof. Dr. Hartmut Pohl zusammen mit einem Seniorberater im Hause des Auftraggebers.

2 Threat Modeling

Threat Modeling unterstützt die methodische Entwicklung eines vertrauenswürdigen Systementwurfs und einer Architektur in der Designphase der Softwareentwicklung – die Fehlerbehebungskosten sind in dieser Entwicklungsphase noch sehr gering. Aus diesem Grund muss das Design eines Softwaresystems unverzichtbar in einem aufwändigen Prozess mit den folgenden 3 Stufen systematisch sicherheitsüberprüft werden:

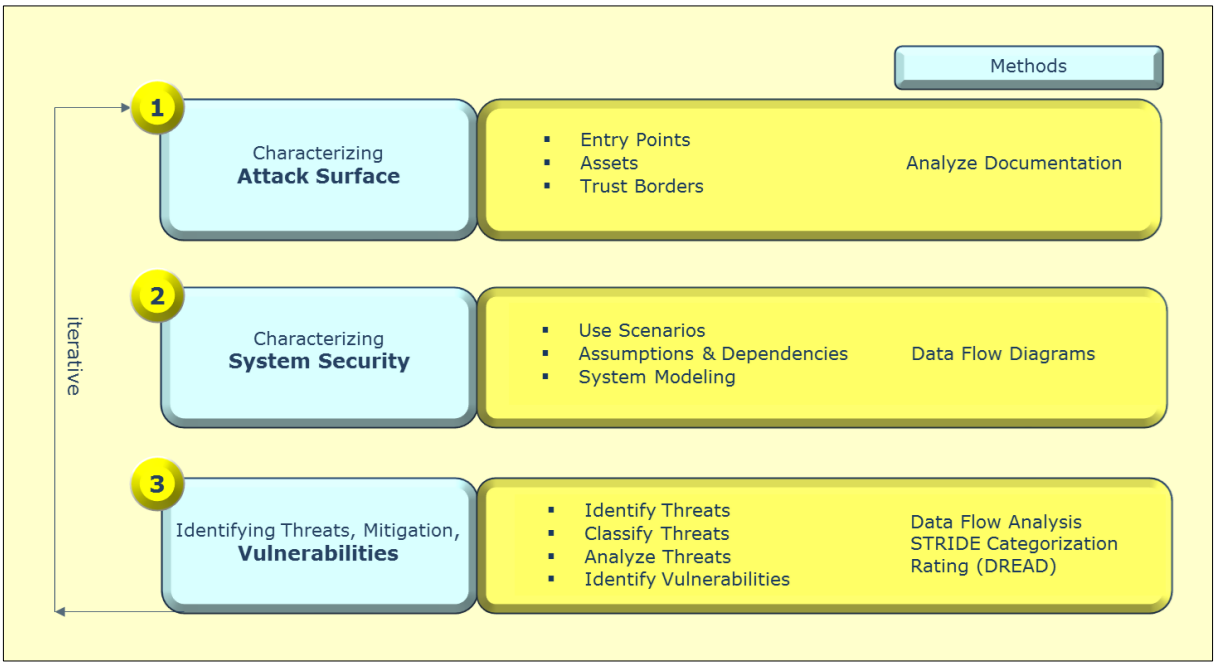

Abb. 2: Threat Model Process

Threat Modeling unterstützt den Aufbau einer vollständigen Sicherheitsarchitektur mit einem angemessenen Sicherheitsniveau und trägt damit dazu bei, die Angriffsfläche des untersuchten Systems zu minimieren.

Die aus dem Threat Modeling gewonnenen Informationen unterstützen den Aufbau einer robusten Sicherheitsarchitektur und tragen dazu bei, die Angriffsfläche des untersuchten Systems zu minimieren.

Die folgenden Arbeitspakete sind Bestandteil unserer Tätigkeiten:

Sichtung und Auswertung verfügbarer Dokumentation des ToE

*ToE = Target of Evaluation. Dazu gehören ALLE Dokumente wie Lastenheft, Pflichtenheft, Programmierdokumentation, Testdokumentation, evtl. Source Code und Programmablaufpläne. Diese Untersuchungen dienen der Erfassung des aktuellen Stands der Entwicklung.

Produktvision, Erstellung von Szenarien

Festlegen des Funktionsumfangs des Threat Model. Unter Berücksichtigung der gesammelten Informationen aus Aktivität 1 wird anhand von Szenarien der Umfang des Threat Models festgelegt.

Diese Szenarien können zur Überprüfung der späteren Implementierung und/oder für Penetration Tests eingesetzt werden.

Szenarien im Smart Grid / Smart Meter Gateway (SMGW) können sein:

- Ein externer Zähler schickt einen Messwert an das SMGW.
- Die Anzeigeneinheit ruft einen Verbrauchswert vom SMGW ab.

Identifizierung der Threat Model Elemente, Entry Points

Die Entry Points werden systematisch aus der vorhandenen Dokumentation identifiziert. Entry Points sind (Eingabe-)Schnittstellen über die auf das ToE zugegriffen werden kann. Ein Entry Point ist nicht nur als physische Schnittstelle zu sehen. Ein einzelner physischer Entry Point kann auch in - mehreren, unterschiedlichen Diensten zugeordnete - logische Entry Points aufgeteilt werden. Nachfolgend ein Beispiel für einen möglichen Entry Point des ToE:

Über die Wide Area Network (WAN)-Schnittstelle kann ein Administrator Tarifierungs- und Konfigurationsdaten oder Software- und Firmware-Updates an das SMGW senden. Der Administrator wird als Trusted Third Party angesehen. Die Kommunikation zwischen SMGW und Administrator muss vertraulich, integer und authentisch erfolgen.

Assets

Die Identifizierung der Assets erfolgt systematisch aus der vorhandenen Dokumentation. Assets stellen die zu schützenden Ressourcen des SMGW dar. Ohne Assets keine Threats: Ein Angreifer versucht, Zu-

griff auf die Assets des SMGW zu erhalten. Bei nicht ausreichenden Sicherheitsmaßnahmen kann ein erfolgreicher Angreifer Assets lesen und/oder schreiben (verändern). Ein Beispiel für ein Asset:
Der Messwert 'aktueller Stromverbrauch' kann so manipuliert werden, dass ein geringerer als der tatsächliche Wert zur Abrechnung übertragen wird.

Trust Level

Die Identifizierung der Trust Level erfolgt systematisch an den Schnittstellen des SMGW mit der Außenwelt. Ein Trust Level charakterisiert eine externe Entität (Benutzer/Technische Komponente) im Hinblick auf Authentifizierung und gewährte Lese-/Schreib-Berechtigungen.

Ein Kunde arbeitet mit einem Kunden-spezifischen, anderen Berechtigungsprofil am SMGW als ein Administrator. Nach Identifizierung der Trust Level können diese den zuvor identifizierten Entry Points und Assets zugeordnet werden. Somit wird ersichtlich, an welchem Entry Point eine externe Entität die Möglichkeit zur Authentifizierung besitzt und welche Lese-/Schreib-Berechtigungen auf Assets gewährt werden.
Die Rolle des externen Marktteilnehmers besitzt dem Bedarf angepasste unterschiedliche Trust Level. So kann ein externer Marktteilnehmer z. B. Verbrauchsdaten abrufen oder nur mit einem CLS-Gerät im HAN kommunizieren.

Vorbereitung Data Flow Diagrams (DFDs)

Zur Vorbereitung der DFDs müssen aus den bisher gesammelten Informationen und der verfügbaren Dokumentation folgende Elemente ermittelt werden:

- External Entities
- Technische Komponenten
- Prozesse
- Data Flows

Erstellung DFDs

Die bisher gesammelten Informationen werden in DFDs umgesetzt. Die Modellierung des gesamten Systems erfolgt dabei in mehreren Ebenen (Level 0, Level 1, ..., Level n+1) und kann von einzelnen technischen Komponenten bis hin zur Darstellung von Funktionsaufrufen durchgeführt werden. Die Tiefe der Modellierung muss vorab festgelegt werden.

Mit den DFDs können die Attack Paths auf das ToE grafisch nachvollzogen werden und bilden somit die Grundlage für die Identifizierung der Threats.

Identifizierung der Threats

Threats werden systematisch aus den gesammelten Informationen und den erstellten DFDs ermittelt werden.

Die identifizierten Threats des SMGW werden mit STRIDE klassifiziert, mit DREAD bewertet und in Threat Trees grafisch dargestellt.

Erstellung von Threat Trees (Attack Trees)

In einem Threat Tree (auch Attack Tree) kann der Zusammenhang der identifizierten Threats grafisch dargestellt werden. Dies dient nicht nur der Übersicht, sondern auch der Kommunikation mit unterschiedlichen - auch weniger technisch orientierten - Personengruppen, die an der Entwicklung des Systems beteiligt sind.

Ein Threat Tree stellt die Angreifbarkeit eines einzelnen Threats dar. Dieser Threat befindet sich in der Wurzel und kann von den Blättern ausgehend angegriffen werden. Die Blätter enthalten mitigierte und unmitigierte Threats.

Ein Attack Path, der von den Blättern ausgehend über unmitigierte Threats bis zur Wurzel nachvollzogen werden kann, stellt eine Sicherheitslücke dar.

Einordnung der Threats in STRIDE

STRIDE dient der Klassifizierung von Threats:

- **S**poofing Identity (Identitätsfälschung)
  Der Angreifer gibt sich als vertrauenswürdige Person oder Programm aus, um Zugriff auf das System zu erhalten.
- **T**ampering with Data (Datenmanipulation)
  Ein nicht autorisierter Angreifer kann Daten manipulieren.
- **R**epudiation (Nicht-Zuordenbarkeit von Ereignissen)
  Aktivitäten, die ein Angreifer am SMGW durchführt, müssen anhand von Protokollen nachgewiesen werden können.
- **I**nformation Disclosure (unberechtigte Weitergabe von Daten)
  Ein Angreifer erlangt Einblick in Daten, für die er keine Leserechte besitzt.
- **D**enial of Service (Einschränkung der Verfügbarkeit)
  Ein Angreifer schränkt die Verfügbarkeit eines laufenden Prozesses ein.

- **E**levation of Privilege (Erweitern von Berechtigungen)
  Ein Angreifer übernimmt einen Account eines bestehenden Benutzers mit limitierten Berechtigungen und versucht höherwertige Berechtigungen zu erlangen.

Nach der Klassifizierung der Threats mit STRIDE werden diese einer ersten Priorisierung unterzogen. So könnten z. B. die Threats der Kategorie Elevation of Privilege bevorzugt zur weiteren Ausarbeitung des Designs berücksichtigt werden.

Mitigating Threats, Einstufung des Risikopotentials identifizierter Threats (DREAD)

Mit DREAD (Damage, Reproducibility, Exploitability, Affected users, Discoverability) wird das Risiko von Threats nach 5 unabhängigen Faktoren rechnerisch ermittelt. Jeder Faktor erhält eine vorab festzulegende Gewichtung. Erfahrungen aus der Praxis zeigen, dass eine möglichst einfache Gewichtung (z. B. von 1-3) effizient von allen Projektbeteiligten eingesetzt werden kann. Die Summe der 5 Faktoren stellt die Risikobewertung von Threats nach DREAD dar. Die Bewertung der Threats erfolgt unabhängig von einer Mitigierung. Nachfolgend ein Beispiel für ein DREAD-Bewertungsschema:

- **D**amage Potential
  Welcher Schaden entsteht bei der Ausnutzung des Threat?
- **R**eproducibility
  Wie leicht ist der Fehler zu reproduzieren?
- **E**xploitability
  Wie leicht kann der Angriff durchgeführt werden?
- **A**ffected Users
- **D**iscoverability
  Wie leicht ist die Sicherheitslücke zu identifizieren?

Nach der Bewertung der Threats können diese in DREAD-Risikoklassen eingeteilt werden. Somit kann die weitere Priorisierung der Threats für die Entwicklung eines sicheren und robusten Designs genutzt werden.

Behebungsstrategien zur Minimierung der Threats (Threat Mitigation)

Die Mitigierung eines Threat erfolgt anhand der verfügbaren Dokumentation. Ein Threat kann dabei in folgende Kategorien eingeteilt werden:

- Mitigiert: Der Threat kann mit Gegenmaßnahmen des aktuellen Entwicklungsstands des SMGW mitigiert werden.
- Teilweise mitigiert: Die Gegenmaßnahmen des aktuellen Entwicklungsstands reichen nicht aus, um einen Threat zu mitigieren.
- Nicht mitigiert: Ein Threat kann mit den derzeit verfügbaren Mitteln nicht mitigiert werden.

Validierung des Threat Model

Nach der Erstellung eines Threat Model muss eine Validierung aller durchgeführten Schritte erfolgen. Nur so können verbliebene Unstimmigkeiten im Threat Model erkannt und beseitigt werden.

Abschlussbericht

Erstellung eines Abschlussberichts unter Bezugnahme auf die konkreten geplanten Maßnahmen zur Erreichung einer überprüften Sicherheitsarchitektur.

Wiederholungsprüfung

Threat Modeling ist ein iterativer Prozess. Die einzelnen Arbeitspakete werden iterativ durchlaufen.

Mit Abschluss der Validierung sollte mit dem Beheben der Sicherheitslücken im aktuellen Design begonnen werden. Durch das Beheben von Sicherheitslücken und den damit verbundenen Änderungen am Design können neue Threats für das System entstehen. Aus diesem Grund ist eine erneute Iteration über alle Arbeitspakete (Wiederholungsprüfung) unverzichtbar.

Präsentation der Ergebnisse

Präsentiert werden alle Ergebnisse von Prof. Dr. Hartmut Pohl zusammen mit einem Seniorberater im Hause des Auftraggebers.

3 Static Source Code Analysis

Dieses Verfahren analysiert den Quellcode, ohne ihn auszuführen (im Gegensatz zur Dynamic Analysis, wozu u.a. Fuzzing zählt). Ab der Implementierungsphase wird die Konformität des Quellcodes der Zielsoftware (White-Box Test!) mit formalen Methoden auf Einhaltung syntaktischer Programmierkonventionen der Programmiersprache und auf Einhaltung der Programmierrichtlinien überprüft - vergleichbar einem Parser, der eine lexikalische, syntaktische und semantische Analyse des Programmcodes durchführt.

Aufgrund lexikalischer Regeln der verwendeten Programmiersprache und den semantischen Zugehörigkeiten benötigen die einzelnen Fehler im Allgemeinen einen manuellen Audit, um false Positives auszuschließen und entsprechende Behebungsstrategien zu entwerfen. Die Qualität und Quantität des Analyse-Resultats hängt somit maßgeblich von der Auswahl geeigneter Tools (und geschultem Fachpersonal) ab.

Verfahren

Static Source Code Analysis (Code Review, statische Source Code Analyse) wird Tool-gestützt automatisiert bzw. semi-automatisiert durchgeführt; die Befunde der Tools werden gesammelt und ‚manuell' ausgewertet. Analysiert wird der Quellcode der Zielsoftware ohne ihn auszuführen (vergl. aber Dynamic Analysis: Fuzzing). Der systematische Ablauf der Static Source Code Analysis ist in Abb. 8 grafisch dargestellt.

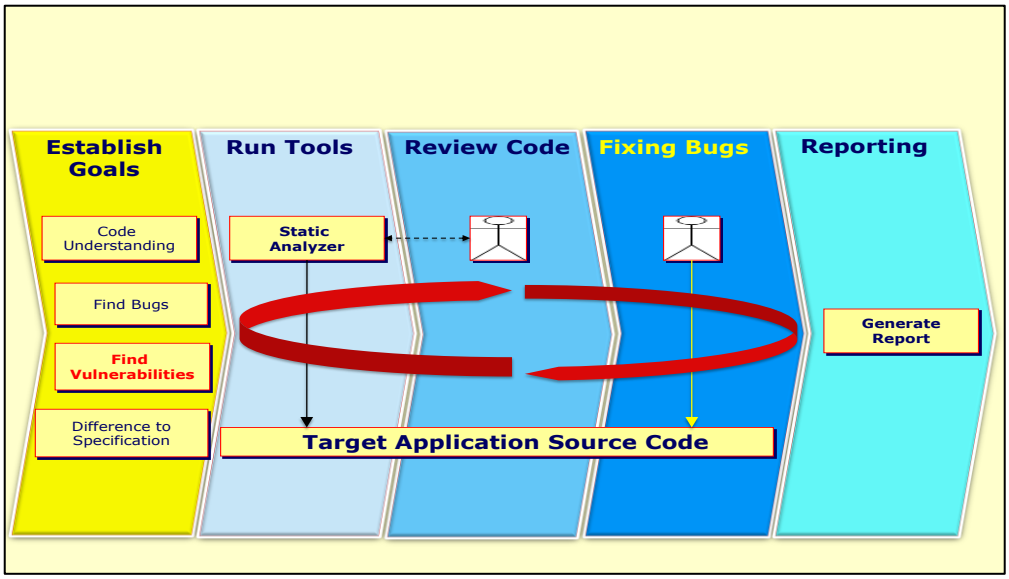

Abb. 3: Static Source Code Analysis Prozess [nach Chess 2007]

Im Bereich Static Source Code Analysis werden drei Tool-Klassen unterschieden:

Style Checking Tools verifizieren den Source-Code bezüglich der Einhaltung syntaktischer Programmier-Richtlinien. Diese einfachen Tools finden meist keine Fehler, die einen Software-Defekt hervorrufen.

Semantic Analysis Tools fügen zusätzliche semantische Informationen zum Syntaxbaum des Compilers hinzu. Diese werden mithilfe verschiedener Regeln auf statisch feststellbare Fehler verifiziert. Typische Fehler sind Datentyp-Probleme, nicht initialisierte Variable und ungenutzte Methoden.

Deep Flow Static Source Code Analysis ist die effektivste Toolklasse: Die semantische Analyse wird um eine „control-flow graph"-Generierung und eine Data Flow Analyse ergänzt. Somit ist es möglich, komplexe Fehler, die etwa auf race conditions, deadlocks oder falscher Pointerverwendung basieren, zu identifizieren.

Die Verwendung eines für das jeweilige Zielsoftware geeignete Static Source Code Analysis Produkt ermöglicht die Identifikation folgender Fehlerarten:

- Uninitialized Variable
- Null Pointer Dereference
- Null Test After Dereference
- Negative Character Value
- Ignored Return Value
- Cast Alters Value
- Unused Value
- Buffer overflows
- Integer Overflow of Alocation Size
- File System Race Condition, Deadlocks, Livelocks
- Memory Leaks
- Double Unlock
- Unreachable Data Flow
- Unreachable Call
- Unreachable Control Flow
- Redundant Condition
- Use After Free
- Negative File Descriptor
- Useless Assignment
- Unreachable Computation
- Double Close
- Security vulnerabilities (z.B. unsichere Funktionen)

Aufgrund lexikalischer Regeln der verwendeten Programmiersprache und semantischen Zugehörigkeiten benötigen die einzelnen Fehler im Allgemeinen einen manuellen Audit durch entsprechendes Fachpersonal (Software-Engineering und IT Security), um false Positives auszuschließen und entsprechende Behebungsstrategien zu entwerfen. Die Qualität und Quantität der Analyse-Resultate hängt somit maßgeblich von der Auswahl geeigneter Tools in Verbindung mit angemessen geschultem Fachpersonal ab.

Mit Static Source Code Analysis lassen sich komplexere Fehler, die beispielsweise auf der Interaktion verschiedener Module beruhen, nur schwer identifizieren; solche Fehler erfordern den Einsatz von Verfahren wie Fuzzing.

Ausgewählte Static Source Code Analysis Tools

Es existiert eine große Menge von Static Source Code Analysis Tools – u.a. Coverity, Findbugs, FxCOP, LAPSE, Pixy (open source), RATS (entgeltfrei), XDepend.

Tools sind grundsätzlich spezialisiert auf eine Programmiersprache; nicht alle Tools identifizieren dieselben Fehler, sodass der Einsatz komplementärer Tools nützlich ist.

Static Source Code Analysis Ergebnisse

In einer Reihe von Projekten wurden pro 1.000 Codezeilen bis zu 50 Vulnerabilities identifiziert – allerdings mussten darüber hinaus (zeitaufwändig!) ca. 30 false positives untersucht werden.

Der Security Analyst muss also die identifizierten Anomalien verifizieren hinsichtlich false Positives und muss mit false Negatives rechnen! False Negatives können noch mit dem Verfahren Dynamic Analysis: Fuzzing identifiziert werden.

Abschlussbericht

Um die Ergebnisse der Static Source Code Analysis festzuhalten und als konkrete Grundlage für weitere Maßnahmen nutzen zu können, werden die identifizierten Vulnerabilities dokumentiert und es werden Gegenmaßnahmen und Handlungsempfehlungen zur Erhöhung der IT-Sicherheit dargestellt.

Wiederholungsprüfung

Mit Abschluss der Static Source Code Analysis sollte mit dem Beheben der Sicherheitslücken im aktuellen Code begonnen werden. Durch das Beheben von Sicherheitslücken und den damit verbundenen Änderungen am Code können neue Sicherheitslücken im Code entstehen. Aus diesem Grund ist eine erneute Static Source Code Analysis (Wiederholungsprüfung) unverzichtbar.

Präsentation der Ergebnisse

Präsentiert werden alle Ergebnisse von Prof. Dr. Hartmut Pohl zusammen mit einem Seniorberater im Hause des Auftraggebers.

4 Penetration Testing Service

Gegenstand ist die Durchführung eines Penetrationstests zur Simulation eines Angriffs auf die vom Kunden genannten Systeme mit Internetanschluss sowie die Analyse und Dokumentation des Angriffs und die Priorisierung und Beschreibung der ermittelten Sicherheitslücken und Abhilfemaßnahmen.
softScheck wird vorhandene, im Handel erhältliche Tools zur Erbringung der Penetration Testing Services verwenden. Diese Tools und die zugehörige Dokumentation verbleiben im Eigentum von softScheck oder des jeweiligen Anbieters.

Die zu untersuchenden externen und internen IP Adressen und die Web Anwendung(en) werden vor Durchführung der Leistungen mit dem Kunden abgestimmt und ausführlich als Anlage zum Vertrag schriftlich dokumentiert.

Wir bieten Ihnen Penetration Testing in 4 wählbaren Modulen an:

Modul 1 - Security Quick Check - Internet Penetration Testing

Ziel der technischen Überprüfung ist die Identifizierung von Sicherheitslücken in Bezug auf IT-Sicherheit auf den IT-Systemen des Auftraggebers. Bei dieser Überprüfung werden ausgewählte Systeme im Internet untersucht.

Für diese Tests werden kommerzielle Tools, wie z. B. der Nessus Vulnerability Scanner, OpenVAS, Nexpose und weitere Tools - je nach Bedarf - eingesetzt.

Modul 2 – Web-Application Penetration Testing

Ziel der technischen Überprüfung ist die Identifizierung von Sicherheitslücken in der vom Auftraggeber eingesetzten Web-Application.

Für diese Tests werden kommerzielle Tools - u.a. der Nessus Vulnerability Scanner, OpenVAS, Nexpose und weitere Tools - je nach Bedarf - eingesetzt.

- Es wird der Server untersucht, auf dem die Web-Application ausgeführt wird.
- Es wird die vom Auftraggeber benannte Web-Application untersucht.

Modul 3 - Network Penetration Testing

Ziel dieser technischen Überprüfung ist es, IT-Systeme mit relevanten Sicherheitslücken, wie z. B. durch Konfigurationsfehler und fehlende Security-Patches zu identifizieren. Zur Durchführung werden kommerzielle Tools – u.a. der Nessus Vulnerability Scanner, aber auch das Exploit-Framework Metasploit - eingesetzt.

Wird das Network Penetration Testing von den Laboren von softScheck durchgeführt, wird zwingend ein VPN-Zugang in das Netzwerk des Auftraggebers benötigt.

Modul 4 Full Penetration Testing

Die Leistungen des Full Penetration umfassen alle Leistungen des:

- Internet Penetration Testing (bis zu 5 IP-Adressen),
- Web-Application Penetration Testing (1 Web-Application) und
- Network Penetration Testing (bis zu 100 IP-Adressen).

Zu jeder Testung gehören die aufgeführten Tätigkeiten:

Projektinitiierung

In dieser Aktivität wird mit den beteiligten Personen die Vorgehensweise und Zielsetzung der Untersuchungen abgestimmt. Hierbei erfolgen u.a. die Festlegung der Überprüfungsschritte, die Definition von Notfallmaßnahmen und die Festlegung sensibler Systeme, die z. B. von der Untersuchung auszuschließen sind.

Network Vulnerability Scans

Das Ziel ist es, relevante Sicherheitslücken, wie z. B. durch Konfigurationsfehler oder fehlende Security-Patches, auf IT-Systemen im Netzwerk des Auftraggebers identifizieren. Es wird empfohlen bei dieser Aktivität möglichst viele IT-Systeme im Netzwerk zu scannen, um realistische Aussagen bezüglich der IT-Sicherheit der Netzwerk-Systeme treffen zu können.

Netzwerkdatenverkehr abhören

Es wird untersucht, an welche Daten und Informationen (z. B. Passwörter, Dokumente, etc.) ein Angreifer im Netzwerk durch das Abhören von Netzwerkdatenverkehr gelangen kann.

Manuelles Verifizieren von identifizierten Sicherheitslücken

Ziel dieser Aktivität ist die Verifikation der zuvor identifizierten Sicherheitslücken. Hierbei werden die identifizierten Sicherheitslücken Tool-basiert oder auch manuell verifiziert, um die tatsächliche oder potenzielle Angreifbarkeit der Systeme und laufenden Dienste festzustellen.

Erstellung des Abschlussberichts

Um die Ergebnisse des Internet und Network Penetration Tests festzuhalten und als konkrete Grundlage für weitere Maßnahmen nutzen zu können, werden die identifizierten Sicherheitslücken dokumentiert und es werden Gegenmaßnahmen und Handlungsempfehlungen zur Erhöhung der IT-Sicherheit dargestellt.

- **Configuration Management**
  - SSL/TLS Check
  - Database Listener Check
  - Infrastructure Configuration Management Check
  - Application Configuration Management Check
  - Repudiation Check
  - Backup/Unreferenced Files Check
  - Admin Interfaces Check
  - HTTP Methods und XST Check
- **Authentication Testing**
  - Credential Transport Check
  - User Enumeration Check
  - Default/Guessable User Accounts Check
  - Authentication Bypass Check
  - Password Remembering/ Reset Check
  - Race Condition Check
- **Authorization Testing**
  - Path Traversal Check
  - Authorization Bypass Check
  - Privilege Escalation Check

Wiederholungsprüfung

Nach den Sicherheitskorrekturen durch den Auftraggeber wird das gesamte Penetration Testing insbesondere unter Berücksichtigung der vorgenommenen Korrekturen erneut geprüft.

Präsentation der Ergebnisse

Präsentiert werden alle Ergebnisse von Prof. Dr. Hartmut Pohl zusammen mit einem Seniorberater im Hause des Auftraggebers.

5 Dynamic Analysis Fuzzing

Gegenstand der Fuzz Testing Services ist die Untersuchung eines vom Kunden benannten Systems auf bislang nicht identifizierte Vulnerabilities sowie die Analyse, Dokumentation und Bewertung der identifizierten Vulnerabilities. Die Dokumentation der identifizierten Vulnerabilities erfolgt priorisiert nach Schweregrad.

softScheck wird vorhandene - kommerzielle und Open Source - Tools zur Erbringung der Fuzz Testing Services verwenden. Diese Tools und die zugehörige Dokumentation verbleiben im Eigentum von softScheck oder des jeweiligen Anbieters.

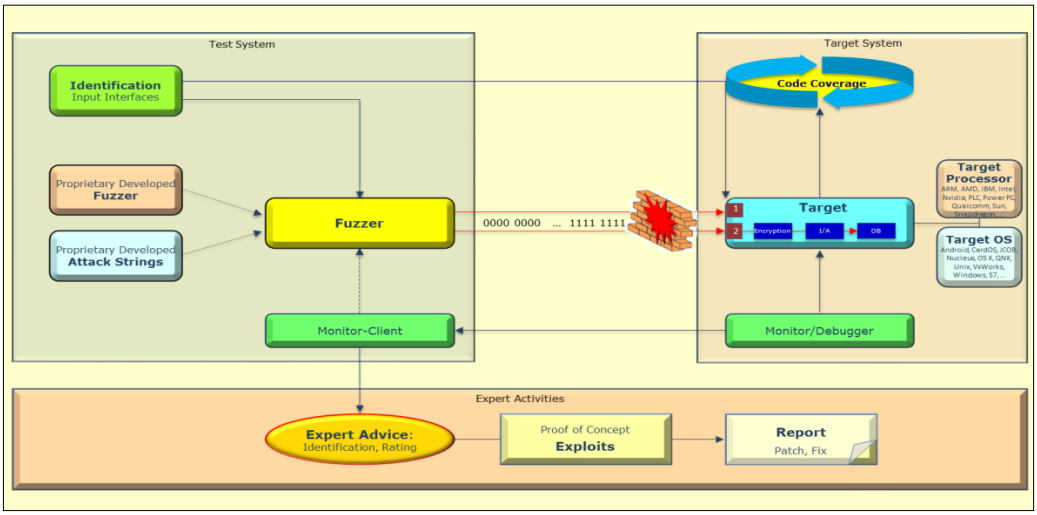

Abb. 4: Fuzzing-Prozess

Fuzz Testing

Für die Untersuchung werden softScheck - falls erforderlich - entsprechende Anmeldeinformationen für das System zur Verfügung gestellt.

Projektinitiierung

Vorgehensweise und Zielsetzung des Fuzz Testing werden mit den Beteiligten die werden abge-stimmt. Hierbei erfolgen u.a. die Definition von Notfallmaßnahmen und die Festlegung sensibler Systeme, die z. B. von der Untersuchung auszuschließen sind.

Die Testumgebung wird vom Auftraggeber zur Verfügung gestellt. Wir gehen von einer fertigen Installation und einer Einweisung in die betriebsfertige Testumgebung aus.

Identifizierung der Schnittstellen

Sofern nicht konkret vorgegeben, werden die Schnittstellen der Zielanwendung identifiziert. Hierbei kann es sich um eine Mensch-zu-Maschine-Schnittstelle (Webinterface, Clientinterface, etc.) oder aber um eine Maschine-zu-Maschine-Schnittstelle (Schnittstellen, die nur für programminterne Abläufe notwendig sind) handeln. Dafür wird – soweit vorhanden – die Dokumentation herangezogen, da die Schnittstellen im Rahmen des technischen Entwurfs festgelegt und deshalb entsprechend dokumentiert sein sollten. Wir gehen davon aus, dass uns dazu ein vollständiger Einblick in die Dokumentation gewährt wird.

Auswahl geeigneter Fuzzing-Tools

Die für die identifizierten Schnittstellen relevanten Fuzzing-Tools werden ausgewählt. Unsere Datenbank mit über 300 evaluierten Fuzzing-Tools unterstützt uns bei der Auswahl der passenden Fuzzing-Tools. Je nach Schnittstelle kommen zwischen 5 und 60 Tools zum Einsatz.

Konfiguration der Fuzzing-Tools auf die Zielanwendung

Ist die Auswahl der Fuzzing-Tools erfolgt, werden die jeweiligen Fuzzing-Tools für die Gegebenheiten der Zielanwendung konfiguriert. Dies ist ein wichtiger Schritt beim Fuzzing-Vorgang, da nur korrekt konfigurierte Tools verlässliche Ergebnisse erzielen.

Monitoring und Auswertung der Monitoring-Ergebnisse

Die Identifizierung von Vulnerabilities erfolgt beim Fuzzing-Prozess durch ein umfassendes Monitoring. Einige Fuzzing-Tools verfügen bereits über eine integrierte Monitoring-Funktionalität, dies reduziert den Arbeitsaufwand beim Fuzz Testing erheblich.

Neben der Überwachung der Zielsoftware können auch ein System-Monitoring und eine so genannte „Valid Case Instrumentation" infrage kommen.

Die vom Monitor gemeldeten Anomalien stellen Hinweise auf tatsächliche Vulnerabilities dar, die verifiziert werden müssen, um false Positives auszuschließen. Hierbei werden die identifizierten Vulnerabilities Tool-basiert oder auch manuell verifiziert, um die tatsächliche oder potenzielle Angreifbarkeit der Systeme und laufenden Dienste festzustellen. Eine notwendige Voraussetzung für eine Vulnerability ist ihre Reproduzierbarkeit sowie das Erreichen eines sicherheitsrelevanten Zustands im Zielprogramm (Buffer Overflow, Code Injection etc.). Der hierzu erforderliche manuelle Aufwand beläuft sich schätzungsweise auf etwa zwei Drittel des Gesamtaufwands zur Identifizierung und Verifizierung von Vulnerabilities.

Es werden zu jeder identifizierten Sicherheitslücke Behebungs- oder Umgehungsmaßnahmen vorgeschlagen.

Bewertung der Vulnerabilities nach CVSSv2

Es ist erforderlich, dass für jede identifizierte Vulnerability eine Bewertung und Priorisierung vorgenommen wird, weil aus Wirtschaftlichkeitsgründen nicht immer alle Vulnerabilities behoben werden.

Die Bewertung der Vulnerabilities wird mit dem „Common Vulnerability Scoring System Version 2.0" (CVSSv2) vorgenommen: Die identifizierten Vulnerabilities werden hinsichtlich ihrer Bedeutung (Severity) bewertet. Dies geschieht anhand einer Vielzahl von Bewertungsparametern, die zum Teil zustandsabhängig sind und auch an die Benutzerumgebung angepasst werden können – dazu gehören:

- Erkennbarkeit für Dritte
- Reproduzierbarkeit
- Ausnutzbarkeit (Exploitability)
- lokal, im Netzwerk und aus dem Internet
- benötigte Zugriffsrechte
- sowie generierbarer Schaden
- Art und Umfang der Auswirkungen auf die Integrität, Verfügbarkeit und Vertraulichkeit der Daten.

Abschlussbericht

Um die Ergebnisse des Fuzz Testing festzuhalten und als konkrete Grundlage für weitere Maßnahmen nutzen zu können, werden die identifizierten Vulnerabilities dokumentiert und es werden Gegenmaßnahmen und Handlungsempfehlungen zur Erhöhung der IT-Sicherheit dargestellt.

Wiederholungsprüfung

Mit Abschluss der Fuzz-Tests sollte mit dem Beheben der Sicherheitslücken im aktuellen Code begonnen werden. Durch das Beheben von Sicherheitslücken und den damit verbundenen Änderungen am Code können neue Sicherheitslücken im Code entstehen. Aus diesem Grund ist ein erneutes Fuzz-Testing (Wiederholungsprüfung) unverzichtbar.

Präsentation der Ergebnisse

Präsentiert werden alle Ergebnisse von Prof. Dr. Hartmut Pohl zusammen mit einem Seniorberater im Hause des Auftraggebers.

6 Konformitätsprüfung

Die Konformitätsprüfung wird in der Designphase im Hinblick auf die Übereinstimmung des Produktdesigns mit den Anforderungen – z.B. der technischen Richtlinie TR 03109 und dem Protection Profile für ein SMGW durchgeführt. Damit wird ein sicheres Produktdesign erreicht.

Vorgehen bei der Konformitätsprüfung:

- Sichtung der Produkt-Dokumentation, Sichtung der technischen Richtlinie und Erstellen einer Prüfliste mit den
- relevanten Prüfkriterien aus der technischen Richtlinie
- Es wird eine Liste der identifizierten Mängel erstellt: Fehlende Übereinstimmung mit der technischen Richtlinie. Die zugehörigen Hinweise auf Fehlerquellen werden ergänzt durch vorgeschlagene Behebungsmaßnahmen.
- Es wird ein Abschlussbericht erstellt
- Nach Behebung der festgestellten Mängel durch den Auftraggeber führt softScheck eine Wiederholungsprüfung durch.
- Zum Abschluss des Projekts erfolgt eine Präsentation der Ergebnisse

Sichtung der Produkt-Dokumentation

Alle verfügbaren Dokumente über das zu prüfende System werden gesichtet. Informationen über den Entwurf werden aus den vom Auftraggeber zur Verfügung gestellten Dokumenten zusammengetragen, systematisch strukturiert und für eine Analyse aufbereitet.

Sichtung der technischen Richtlinie und des Protection Profiles

Alle relevanten Informationen der technischen Richtlinie und des Protection Profiles der verantwortlichen Zertifizierungsinstitution (BSI) werden mit Hinblick auf das zu prüfende System gesichtet und auf Aktualität geprüft. Abgestimmt auf die Kundenanforderungen werden die wesentlichen Vorgaben extrahiert.

Erstellen einer Prüfliste

Alle gesammelten Prüfkriterien des Schutzprofils werden kategorisiert und zu einer Prüfliste zusammengestellt.

Durchführung der Konformitätsprüfung: Kommunikation mit WAN

Jede Kommunikation zwischen SMGW und WAN wird auf das Einhalten der sicherheitstechnischen Anforderungen überprüft. Dazu zählen die Bereiche:

- Korrekte Nutzung von Transport Layer Security (TLS)
- Schlüssel-/Zertifikatsmanagement
- Firmware-Updates/Downloads
- Wake-Up Service
- SMGW-Monitoring
- Zeitsynchronisation
- Auslieferung von tarifierten Messwerten oder Netzzustandsdaten
- Alarmierungen und Benachrichtigungen
- Kommunikation zwischen externen Marktteilnehmern und Controllable Local Systems (CLS)

Konformitätsprüfung: Kommunikation mit LMN

Jede Kommunikation zwischen SMGW und LMN wird auf das Einhalten der sicherheitstechnischen Anforderungen überprüft. Dazu zählen die Bereiche:

- Korrekte Nutzung von TLS
- Registrierung und Konfiguration von Zählern
- Schlüssel-/Zertifikatsmanagement

Konformitätsprüfung: Kommunikation mit HAN

Jede Kommunikation zwischen SMGW und HAN wird auf das Einhalten der sicherheitstechnischen Anforderungen überprüft. Dazu zählen die Bereiche:

- Korrekte Nutzung von TLS
- Kommunikationskanäle zwischen externer Marktteilnehmer und CLS
- Identifizierung und Authentisierung

Konformitätsprüfung: Zusammenspiel zwischen SMGW und HSM

Die Kommunikation zwischen SMGW und HSM muss den sicherheitstechnischen Anforderungen genügen. Dazu zählt vor allem die korrekte Nutzung von Verschlüsselungs- und Authentisierungsinformationen:

- Nutzung von HSM-Daten bei Anwendung des TLS-Handshakes
- Nutzung von HSM-Daten bei der CMS Inhaltssicherung

Konformitätsprüfung: Public Key Infrastructure

Die Architektur der Public Key Infrastructure (PKI) muss den Requirements der PKI-Richtlinie entsprechen. Darunter fallen die folgenden Bereiche:

- Verwendung der Zertifikate
- Zertifikatsablauf
- Erneuern von Zertifikaten
- Nutzung von Sperrlisten

Auswerten der Prüfungsergebnisse: Validierung aufgetretener Mängel

Jeder festgestellte Mangel wird einer Zweitprüfung unterzogen, um False Positives auszuschließen.

Hinweise auf Fehlerquellen und Behebungsmaßnahmen

Es werden - sofern erforderlich - Behebungs- oder Umgehungsmaßnahmen vorgeschlagen.

Erstellen des Abschlussberichts zu den identifizierten Konformitätsmängeln

Der Abschlussbericht beinhaltet die identifizierten Mängel sowie Vorschläge zu entsprechenden Behebungsstrategien und nimmt eine Bewertung des Sicherheitsniveaus des Produktdesigns basierend auf der Technischen Richtlinie vor. Es werden alle für den Auftraggeber relevanten Daten in diesem Dokument zusammengefasst und festgehalten.

Wiederholungsprüfung aller Arbeitspakete

Nach den Fehlerkorrekturen wird das Gesamtsystem insbesondere unter Berücksichtigung der vorgenommenen Korrekturen erneut geprüft.

Präsentation der Ergebnisse

Präsentiert werden alle Ergebnisse von Prof. Dr. Hartmut Pohl zusammen mit einem Seniorberater im Hause des Auftraggebers.

Prof. Dr. Hartmut Pohl
Geschäftsführender Gesellschafter
soft5check GmbH Köln www. soft5check.com

Büro: Bonner Str. 108. 53757 Sankt Augustin

Tel.: +49 (2241) 255 43 - 0

Mobil: +49 (172) 9437 - 329
Fax: +49 (2241) 255 43 - 29

Hartmut.Pohl@soft5check.com