Application Security Audits in der Praxis - Prüfstein oder Schikane?

Christoph Baumgartner

Agenda

- Verbreitete Irrtümer
- Ausgangslage
- Definitionen
- Zweck und Nutzen
- Tasks
- Projektetappen
- Angewandte Methoden
- Mengengerüst
- Applikatorische Sicherheit
- Techniken
- Findings
- Fazit
- Empfehlungen
- Kosten und Aufwand
- Beantwortung von Fragen
- Beilage OSSTMM

Verbreitete Irrtümer

- Bei einem Application Security Audit muss nur die eigentliche Applikation getestet werden
- Nur was angeklickt werden kann, kann auch ausgeführt werden
- Funktionalität geht vor Sicherheit
- Applikationshersteller unternehmen alles, um ihre Applikationen sicher zu gestalten

Ausgangslage

- Privatbank mit mehrere Niederlassungen im Ausland
- CRM-System selbständig weiterentwickelt (Multi-tier Application)
- Sicherheitsüberprüfung als Qualitätssicherungs-massnahme vor Roll-out der Lösung in Nieder-lassungen
- OneConsult wurde mit *Application Security Audit* beauftragt

Definition Application Security Audit

Ganzheitliche Applikations-Sicherheitsüberprüfung

- Technische Aspekte
  - Infrastruktur
  - Vertrauensstellungen
  - Sicherheitslücken
  - etc.
- Organisatorische Aspekte
  - Verantwortlichkeiten
  - Dokumentationspflege
  - Umgang mit Source Code
  - Patchingprozess
  - etc.

Definitionen Angreifertypen

- Ein **User** ist ein Anwender, welcher über das nötige Wissen verfügt, um die **Informatikmittel funktionsgerecht zu nutzen**.
- Als «**Skript Kiddie**» wird eine Person bezeichnet, welche über (relativ) **wenig Computer- und Netzwerkfachwissen** verfügt, aber unbedarft (vollautomatisierte) **Hackertools einsetzt**.
- Ein «**Hacker**»/«**Cracker**» **bricht ohne Erlaubnis** des Systemeigners meist via Computernetze **in Computersysteme ein** oder knackt das Lizenzsystem von Computerprogrammen. Dafür umgeht er bewusst Sicherheitsmechanismen. Die Beweggründe sind Ehrgeiz, möglicher finanzieller Profit, Geltungssucht, Idealismus oder Zerstörungswille.
- Ein «**Ethical Hacker**» ist ein Computer- und Netzwerkspezialist, welcher **im Auftrag des Systemeigners** nach **Systemverwundbarkeiten sucht**, welche ein «Hacker»/«Cracker» ausnutzen könnte.
- Ein «**Social Engineer**» versucht mittels Ausnutzens menschlicher Schwächen an vertrauliche Daten zu gelangen.

Zweck und Nutzen

- **Qualitätssicherung** dank (unabhängiger) IT Security-Analyse
- **Compliancy**-Nachweis bezüglich gesetzlicher Rahmenbedingungen und Vorgaben
- **Prävention** ermöglicht direkte und indirekte Kosteneinsparungen (in der Zukunft)
- **Awareness** Building auf allen Stufen und Know-how Transfer
- **Argumentationsgrundlage** für zukünftige IT Security-Projekte bzw. -Aktivitäten

Tasks: Teil 1

- Dokumentenbezogener Design- und Konzept-Review
  - Berechtigungskonzept
  - Auditing / Logging
  - Datenverfremdung
  - Netzwerktopologie und Systemarchitektur
  - Authentifizierung und Verschlüsselung (Richtlinien, Schnittstellen, Risiken, etc.)
  - Backup Prozess

Tasks: Teil 2

- Configuration Review
  - Systemparameter, Dateisystem, Patchlevel der Betriebssysteme und Applikationen
  - Berechtigungen auf Netzwerk-, Dateisystem- und Benutzerebene
  - Datenbank-Listener
  - Firewall-Konfiguration und -Regeln
  - Applikationslogin vs. Datenbanklogin
  - Verfügbarkeit und Wartung
  - Datenbank: Schemata, Rollen, Rechte
  - Schnittstellen und Kommunikation
- Penetration Test Untersuchungsobjekt

Out of Scope

Auf folgende Techniken wurde explizit verzichtet:

- Code Review
- Denial-of-Service (DoS)
- Informationsbeschaffung mittels
  - Social Engineering
  - Dumpster Diving

Projektetappen

- Kick-off Meeting
- Dokumentationsanalyse
- Penetration Test:
  - Zeigt Schwachstellen der verwendeten Komponenten auf
  - Basis für applikatorische Sicherheitsüberprüfung
- Application Security Audit: zeigt applikationsbedingte Sicherheitslücken auf
- Schlussbericht
- Präsentation/Diskussion

Angewandte Methoden

- Penetration Test: *Open Source Security Testing Methodology Manual* (OSSTMM) ist hinsichtlich technischer Audits kompatibel zu ISO 17799, IT Grundschutzhandbuch und SOX
- Application Security Audit: *Secure Programming Standards Methodology Manual* (SPSMM)
- White-box-Approach (Offenlegung des Untersuchungsobjekts)
- Iterativer Testzyklus
  - Testen
  - Sicherheitslücken schliessen
  - Erneut testen (Nachkontrolle)

Mengengerüst Untersuchungsobjekt

Untersuchungsobjekt (im LAN):

- 1 Server mit Test-Datenbank
- 1 PC-Arbeitsplatz mit «Doppelfunktion»
  - Client-Applikation
  - Zielsystem für «Brückenkopf»-Angriffe
- 1 Firewall

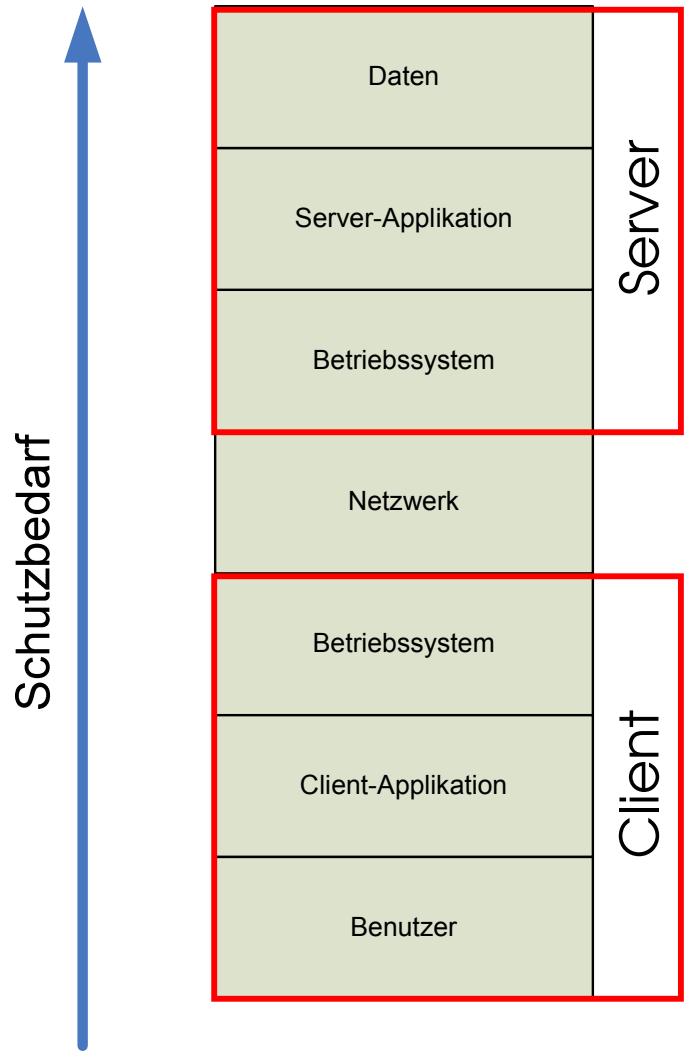
Applikatorische Sicherheit: Teil 1

- Applikation gilt als sicher, wenn keine involvierte Komponente dazu missbraucht werden kann, die definierten Sicherheitsziele negativ zu beeinträchtigen

  ➡ Applikation kann höchstens so sicher sein, wie das Umfeld, in welchem sie betrieben wird

  - Betriebssysteme (Client, Server und zwischengeschaltete Netzwerkkomponenten)
  - Webserver
  - Datenbanken
  - etc.

Applikatorische Sicherheit: Teil 2



- Jede Komponente (Hard- und Software) soll so abgesichert sein, dass sie die Sicherheitsanforderungen der darüber liegenden Komponente erfüllt.

  Beispiele:

  - Benutzer kann nicht direkt mit Betriebssystem(diensten) interagieren
  - Netzwerk lässt nur Kommunikation von/mit zugelassenen Clients zu

- Generell gilt Misstrauensprinzip

Techniken (Auswahl): 1. Teil

Technik	Beschreibung	Zweck
Sichtung der Dokumentation	▫ Analyse Applikations- und Datenbankdesign ▫ Analyse Konzeptionsvorgaben	SOLL-Vorgaben eruieren
Input Validation	▫ Bewusster «Missbrauch» der Eingabefelder der Client-Applikation durch Eingabe von Malicious Code-Sequenzen (z.B. Buffer Overflow oder SQL Injection)	Prüfen, ob sich die Applikation in einen undefinierten Zustand bringen lässt oder ob der Programmverlauf oder das -verhalten verändert werden kann
Netzwerk-Sniffing	▫ Mitschnitt und Analyse des Netzwerkverkehrs zwischen verschiedenen Komponenten	Prüfen, ob sich (sicherheits-) relevante Informationen (User-ID, Passwort oder vertrauliche Daten) erlangen lassen

Techniken (Auswahl): 2. Teil

Technik	Beschreibung	Zweck
API Monitoring	▫ Analyse des Datenverkehrs zwischen Applikation und Betriebs-systemkomponenten	Suche nach Schwach-stellen in interner Kommuni-kation
Reverse Engineering / Debugging	▫ Analyse von Systemaufrufen (z.B. String-Manipulationen oder Verschlüsselungsaufrufe) im lauffähigen Programm	Suche nach systemaufruf- und applikationsbasierten Sicherheitslücken
Decompilation	▫ Versuch, aus dem kompilierten Programm den zu Grunde liegen-den (Human readable) Source Code zu regenerieren	Konvertierung zwecks vereinfachter Suche im Source Code nach Schwachstellen
Code Review (war nicht Teil des Projekts)	▫ Analyse des Original-Source Codes	Suche nach Schwach-stellen im Source Code

Findings

- Definierte Funktionalität (Manual) entspricht grösstenteils der implementierten (Applikation)
- Auswahl Sicherheitslücken:
  - Daten lassen sich via Print Screen «exportieren»
  - API Monitoring ermöglicht Unterwanderung der verschlüsselten Kommunikation zwischen Client und Server
  - Standard-Datenbankuser mit Defaultpasswort
  - Datenbank-Listener anfällig auf Buffer Overflow-Attacken
  - Client-Debug-Modus erlaubt Rückschlüsse auf Applikationsdesign
  - Server akzeptiert Source Routing
  - Direkte Kommunikation mit Datenbank möglich
  - Passwort-Policy für lokale Benutzer auf Server nicht umgesetzt

Fazit

- Erhöhte (technische) Sicherheit: Sicherheitsniveau des Untersuchungsobjekts während des Projekts kontinuierlich gesteigert
  - Betriebssysteme
  - Eigentliche Applikation
  - Drittapplikationen
- Gesteigerte Awareness: Coaching der involvierten Mitarbeiter des Kunden und Standpunkt «durch die Brille eines Hackers» geschätzt
- «Hausaufgaben»: Nach Projektabschluss verbleibende Sicherheitslücken (soweit sinnvoll und möglich) systematisch geschlossen; als Folge Roll-out-Termin um 3 Monate nach hinten verschoben

 Projektziele erreicht, anspruchsvoller Prüfstein

Systembezogene Empfehlungen: Teil 1

- Client-Applikation primär für Darstellung und Bearbeitung von Daten zuständig
- Zugriffsrelevante Funktionen ausschliesslich Server-Komponenten-gesteuert (zentraler Ansatz)
- Server-Komponente sollte auch einem versierten Angreifer trotzen können, der bereits im Besitz eines gültigen User-Logins mit eingeschränkter Berechtigung ist
- Kommunikation zwischen Komponenten verschlüsseln
- Auditing (Logging) sämtlicher Aktivitäten
- Für Audits verfremdete Test-Daten bereitstellen

Systembezogene Empfehlungen: Teil 2

- Defaultaccounts und -passwörter (z.B. bei Installation) generell setzen/ändern
- Starke Passwörter (mind. 8 Zeichen, komplex) und regelmässigen Passwortwechsel erzwingen
- Nicht benötigte User-Accounts und Dienste
  - Phase 1: deaktivieren
  - Phase 2: deinstallieren
- Bedürfnisgerechte User-Rechte vergeben
- Schriftlich formulierte Sicherheitsrichtlinien (Security Policies) einführen/pflegen
- System-Verantwortungen und -Kompetenzen klar regeln
- Regelmässiges Security-Patching ausführen
- Regelmässigen System-Reboot durchführen
- Banner-Spoofing einrichten
- Regelmässige Nachkontrollen durchführen

Generelle Empfehlungen

- Management Attention bzw. Projektleitung durch Auftraggeber
- Teamwork sichert Projekterfolg
- Präzise Projektplanung und -durchführung:
  - Rahmenbedingungen (inkl. Kontrolle über deren Einhaltung)
  - Rollendefinition (inkl. Notfallplan)
  - Protokollierung jeglicher Testaktivitäten / Mitschnitt des Netzwerkverkehrs während Tests
  - Mitarbeiterinformation (?)
  - etc.
- White-Box-Approach (Offenlegung des Untersuchungsobjekts) anstatt Black-Box-Approach
- Positive Aspekte würdigen
- Nachvollziehbare Methodik des Auftragnehmers

Kosten und Aufwand

- Zeitaufwand OneConsult in diesem Projekt: 14 Personentage
- Externe Kosten (bei seriöser Durchführung):
  - Minimum: CHF 12`000
  - Maximum: offen
  - Durchschnitt: CHF 25`000 – 40`000
- Zeitaufwand seitens Auftraggeber bzw. Betreuer:
  - Minimum: 30 Minuten pro Testtag
  - Besser: Bereitschaftsdienst Kontaktperson und Systemadministratoren während der Durchführung aller Tests
  - Optional: gemeinsame Arbeit mit Auftragnehmer
  - Zusätzlich: Umsetzung Massnahmen

Beantwortung von Fragen

Besten Dank für Ihr Interesse!
Gerne beantworte ich Ihre Fragen:

Christoph Baumgartner

lic. oec. publ., OPST
CEO / Senior Consultant

info@oneconsult.com
+41 (0)79 421 20 01

OneConsult®

OneConsult GmbH
Zürcherstrasse 73
8800 Thalwil
Schweiz

http://www.oneconsult.com
info@oneconsult.com
Tel. +41 (0)43 443 52 52
Fax +41 (0)43 443 52 62

…mit Sicherheit bessere Lösungen

OSSTMM

OSSTMM (Open Source Security Testing Methodology Manual, von ISECOM), http://www.osstmm.org, seit 2001

- Offene Methodik zur Planung, Durchführung und Dokumentation technischer Sicherheitsüberprüfungen
- Hinsichtlich technischer Audits kompatibel zu ISO 17799, IT Grundschutzhandbuch und SOX
- Verhaltenskodex für Tester
- Manual beinhaltet Methodik und Formulare
- Verschiedene Zertifizierungsmöglichkeiten
  - OPSE (OSSTMM Professional Security Expert)
  - OPST (OSSTMM Professional Security Tester)
  - OPSA (OSSTMM Professional Security Analyst)