Bundesamt
für Sicherheit in der
Informationstechnik

APP.4.6 SAP ABAP-Programmierung

1. Beschreibung

1.1. Einleitung

Häufig werden in SAP-Systemen Eigenentwicklungen programmiert. Die Gründe dafür sind vielfältig, so können Geschäftsprozesse oder Anforderungen an das Reporting mit Hilfe von Eigenentwicklungen individuell an die Institution angepasst werden. Außerdem ist es möglich, spezielle Funktionen zu erstellen, die in der Standard-Auslieferung nicht vorhanden sind.

Eigenentwicklungen werden von Entwickelnden der Institution oder von beauftragten Entwickelnden programmiert. Im SAP-Umfeld wird dazu häufig ABAP (Advanced Business Application Programming) verwendet.

ABAP ist eine proprietäre, plattformunabhängige Programmiersprache des Unternehmens SAP. Sie wurde für die Programmierung kommerzieller Anwendungen im SAP-Umfeld entwickelt und ähnelt in ihrer Grundstruktur entfernt der Sprache COBOL. Wichtige Merkmale sind:

- Integration eines Authentisierungs-, Rollen- und Berechtigungskonzepts,
- Verwendung eines proprietären, datenbankunabhängigen SQL-Derivats (Open SQL),
- Unterstützung der Kommunikation zwischen verschiedenen SAP-Systemen sowie
- Integration von Audit-Optionen.

1.2. Zielsetzung

Der Baustein zeigt ABAP-Entwickelnden und Sicherheitstestenden relevante technische Risiken auf, die sich durch ABAP-Eigenentwicklungen ergeben können. Außerdem werden Anforderungen definiert, die aufzeigen, wie ABAP-Programme sicher entwickelt und eingesetzt werden können.

Der Baustein setzt grundlegende Kenntnisse in ABAP und im Umgang mit ABAP-Entwicklungswerkzeugen voraus.

1.3. Abgrenzung und Modellierung

Der Baustein APP.4.6 *SAP ABAP-Programmierung* ist auf jedes SAP-System einmal anzuwenden, wenn Eigenentwicklungen in der Programmiersprache ABAP erstellt werden.

Mit diesem Baustein werden die Bausteine CON.8 *Software-Entwicklung*, APP.6 *Allgemeine Software* und APP.7 *Entwicklung von Individualsoftware* um konkrete Aspekte zur Entwicklung von ABAP-Programmen erweitert.

Der Baustein stellt keine vollständige Anleitung dar, um ABAP-Programme zu entwickeln, sondern beschreibt die generellen Risiken der Programmiersprache ABAP. Im Baustein werden Anforderungen definiert, die bei der Entwicklung von ABAP-Programmen aus Sicherheitssicht erfüllt werden sollten.

Da Webanwendungen nur einen sehr geringen Anteil aller ABAP-Anwendungen in SAP-Implementierungen ausmachen, stehen Web-Schwachstellen nicht im Fokus dieses Dokuments.

2. Gefährdungslage

Da IT-Grundschutz-Bausteine nicht auf individuelle Informationsverbünde eingehen können, werden zur Darstellung der Gefährdungslage typische Szenarien zugrunde gelegt. Die folgenden spezifischen Bedrohungen und Schwachstellen sind für den Baustein APP.4.6 *SAP ABAP-Programmierung* von besonderer Bedeutung.

2.1. Fehlende Berechtigungsprüfungen

In SAP werden Berechtigungen nur dann geprüft, wenn eine entsprechende Berechtigungsprüfung von Entwickelnden im Programm implementiert wurde. Ohne eine solche Prüfung im Programm-Code wird also nicht getestet, ob Benutzende auch wirklich berechtigt sind eine Aktion auszuführen. In selbst entwickeltem Programm-Code werden Berechtigungsprüfungen aber häufig vergessen. Somit greift das gesamte Berechtigungskonzept oftmals nicht und unberechtigte Personen können auf die im SAP-System gespeicherten Daten zugreifen. Dadurch kann etwa auch gegen Compliance-Anforderungen verstoßen werden. Dies kann besonders bei Wirtschaftsprüfungen schwerwiegende Folgen nach sich ziehen.

2.2. Verlust von Vertraulichkeit oder Integrität von kritischen Daten

SAP-Systeme enthalten viele institutionskritische Informationen. Der SAP-Standard sieht verschiedene Mechanismen vor, diese Daten zu schützen. Allerdings könnte durch fehlerhafte ABAP-Eigenentwicklungen unerlaubt auf institutionskritische Informationen zugegriffen werden. Mitarbeitende oder Angreifende könnten die Daten so in eine nicht mehr kontrollierbare Umgebung transferieren. Ebenso könnten mit Hilfe von ABAP-Programmen kritische Daten manipuliert werden, indem die Sicherheitsmechanismen des SAP-Standards umgangen werden.

2.3. Injection-Schwachstellen

Injection-Schwachstellen entstehen dadurch, dass Angreifende Steuerzeichen bzw. Kommandos über das Eingabefeld in eine Anwendung einschleust. Ein erfolgreicher Angriff kann den geplanten Programmablauf durch unerwartete Kommandos stören.

Injection-Schwachstellen stellen für Eigenentwicklungen das größte Sicherheitsrisiko dar. Durch fehlerhaften Code in einer ABAP-Anwendung können Angreifende ein SAP-System mitunter vollständig kontrollieren. Da solche Angriffe sehr komplex sind und es viele Varianten davon gibt, lassen sie sich ohne spezielle Schulungen kaum erkennen und beheben.

2.4. Umgehung von vorhandenen SAP-Sicherheitsmechanismen

Der SAP-Standard stellt verschiedene Schutzmechanismen für Daten zur Verfügung. Dazu gehören unter anderem die Mandantentrennung, Identity-Management sowie Rollen und Berechtigungen. Diese Sicherheitsmechanismen können im Code jedoch bewusst umgangen oder ungewollt weggelassen werden.

3. Anforderungen

Im Folgenden sind die spezifischen Anforderungen des Bausteins APP.4.6 *SAP ABAP-Programmierung* aufgeführt. Der oder die Informationssicherheitsbeauftragte (ISB) ist dafür zuständig, dass alle Anforderungen gemäß dem festgelegten Sicherheitskonzept erfüllt und überprüft werden. Bei strategischen Entscheidungen ist der oder die ISB stets einzubeziehen.

Im IT-Grundschutz-Kompendium sind darüber hinaus weitere Rollen definiert. Sie sollten besetzt werden, insofern dies sinnvoll und angemessen ist.

Zuständigkeiten	Rollen
Grundsätzlich zuständig	Entwickelnde
Weitere Zuständigkeiten	Keine

Genau eine Rolle sollte *Grundsätzlich zuständig* sein. Darüber hinaus kann es noch *Weitere Zuständigkeiten* geben. Falls eine dieser weiteren Rollen für die Erfüllung einer Anforderung vorrangig zuständig ist, dann wird diese Rolle hinter der Überschrift der Anforderung in eckigen Klammern aufgeführt. Die Verwendung des Singulars oder Plurals sagt nichts darüber aus, wie viele Personen diese Rollen ausfüllen sollen.

3.1. Basis-Anforderungen

Die folgenden Anforderungen MÜSSEN für diesen Baustein vorrangig erfüllt werden:

APP.4.6.A1 Absicherung von Reports mit Berechtigungsprüfungen (B)

Es MUSS sichergestellt sein, dass nur berechtigte Personen selbst programmierte Auswertungen (Reports) starten können. Deswegen MUSS jeder Report explizite, zum Kontext passende Berechtigungsprüfungen durchführen.

APP.4.6.A2 Formal korrekte Auswertung von Berechtigungsprüfungen (B)

Jede Berechtigungsprüfung im Code MUSS durch Abfrage des Rückgabewertes *SY-SUBRC* ausgewertet werden.

APP.4.6.A3 Berechtigungsprüfung vor dem Start einer Transaktion (B)

Wenn Entwickelnde den Befehl *CALL TRANSACTION* verwenden, MUSS vorher immer eine Startberechtigungsprüfung durchgeführt werden.

APP.4.6.A4 Verzicht auf proprietäre Berechtigungsprüfungen (B)

Jede Berechtigungsprüfung MUSS technisch über den dafür vorgesehenen Befehl *AUTHORITY-CHECK* erfolgen. Proprietäre Berechtigungsprüfungen, z. B. basierend auf Konto-Kennungen, DÜRFEN NICHT benutzt werden.

3.2. Standard-Anforderungen

Gemeinsam mit den Basis-Anforderungen entsprechen die folgenden Anforderungen dem Stand der Technik für diesen Baustein. Sie SOLLTEN grundsätzlich erfüllt werden.

APP.4.6.A5 Erstellung einer Richtlinie für die ABAP-Entwicklung (S)

Es SOLLTE eine Richtlinie für die Entwicklung von ABAP-Programmen erstellt werden. Die Richtlinie SOLLTE neben Namenskonventionen auch Vorgaben zu ABAP-Elementen beinhalten, die verwendet bzw. nicht verwendet werden dürfen. Die Anforderungen aus diesem Baustein SOLLTEN in die Richtlinie aufgenommen werden. Die Richtlinie SOLLTE für die Entwickelnden verbindlich sein.

APP.4.6.A6 Vollständige Ausführung von Berechtigungsprüfungen (S)

Bei einer Berechtigungsprüfung im ABAP-Code (*AUTHORITY-CHECK <OBJECT>*) SOLLTE sichergestellt sein, dass alle Felder des relevanten Berechtigungsobjekts überprüft werden. Wenn einzelne Felder tatsächlich nicht benötigt werden, SOLLTEN sie als *DUMMY* gekennzeichnet werden. Zusätzlich SOLLTE am Feld der Grund für die Ausnahme dokumentiert werden.

APP.4.6.A7 Berechtigungsprüfung während der Eingabeverarbeitung (S)

Funktionscodes und Bildschirmelemente von ABAP-Dynpro-Anwendungen SOLLTEN konsistent sein. Wenn ein Bildschirmelement abgeschaltet wurde, dann SOLLTE eine Anwendung NICHT ohne adäquate Berechtigungsprüfungen auf Ereignisse dieses Elements reagieren. Wenn bestimmte Einträge eines Dynpro-Menüs ausgeblendet oder einzelne Schaltflächen deaktiviert werden, dann SOLLTEN auch die zugehörigen Funktionscodes nicht ausgeführt werden.

APP.4.6.A8 Schutz vor unberechtigten oder manipulierenden Zugriffen auf das Dateisystem (S)

Wenn Zugriffe auf Dateien des SAP-Servers von Eingaben der Benutzenden abhängen, SOLLTEN diese Eingaben vor dem Zugriff validiert werden.

APP.4.6.A9 Berechtigungsprüfung in remote-fähigen Funktionsbausteinen (S)

Es SOLLTE sichergestellt werden, dass alle remote-fähigen Funktionsbausteine im Programmcode explizit prüfen, ob der Aufrufende berechtigt ist, die zugehörige Businesslogik auszuführen.

APP.4.6.A10 Verhinderung der Ausführung von Betriebssystemkommandos (S)

Jedem Aufruf eines erlaubten Betriebssystemkommandos SOLLTE eine entsprechende Berechtigungsprüfung (Berechtigungsobjekt *S_LOG_COM*) vorangestellt werden. Eingaben von Benutzenden SOLLTEN NICHT Teil eines Kommandos sein. Deswegen SOLLTEN Betriebssystemaufrufe ausschließlich über dafür vorgesehene SAP-Standardfunktionsbausteine ausgeführt werden.

APP.4.6.A11 Vermeidung von eingeschleustem Schadcode (S)

Die ABAP-Befehle *INSERT REPORT* und *GENERATE SUBROUTINE POOL* SOLLTEN NICHT verwendet werden.

APP.4.6.A12 Vermeidung von generischer Modulausführung (S)

Transaktionen, Programme, Funktionsbausteine und Methoden SOLLTEN NICHT generisch ausführbar sein. Sollte es wichtige Gründe für eine generische Ausführung geben, SOLLTE detailliert dokumentiert werden, wo und warum dies geschieht. Zusätzlich SOLLTE eine Allowlist definiert werden, die alle erlaubten Module enthält. Bevor ein Modul aufgerufen wird, SOLLTE die Eingabe von Benutzenden mit der Allowlist abgeglichen werden.

APP.4.6.A13 Vermeidung von generischem Zugriff auf Tabelleninhalte (S)

Tabelleninhalte SOLLTEN NICHT generisch ausgelesen werden. Sollte es wichtige Gründe dafür geben, dies doch zu tun, SOLLTE detailliert dokumentiert werden, wo und warum dies geschieht. Außerdem SOLLTE dann gewährleistet sein, dass sich der dynamische Tabellenname auf eine kontrollierbare Liste von Werten beschränkt.

APP.4.6.A14 Vermeidung von nativen SQL-Anweisungen (S)

Die Schnittstelle ABAP Database Connectivity (ADBC) SOLLTE NICHT verwendet werden. Eingaben von Benutzenden SOLLTEN NICHT Teil von ADBC-Befehlen sein.

APP.4.6.A15 Vermeidung von Datenlecks (S)

Es SOLLTE eine ausreichend sichere Berechtigungsprüfung durchgeführt werden, bevor geschäftskritische Daten angezeigt, übermittelt oder exportiert werden. Vorgesehene (gewollte) Möglichkeiten des Exports SOLLTEN dokumentiert werden.

APP.4.6.A16 Verzicht auf systemabhängige Funktionsausführung (S)

ABAP-Programme SOLLTEN NICHT systemabhängig programmiert werden, so dass sie nur auf einem bestimmten SAP-System ausgeführt werden können. Sollte dies jedoch unbedingt erforderlich sein, SOLLTE es detailliert dokumentiert werden. Außerdem SOLLTE der Code dann manuell überprüft werden.

APP.4.6.A17 Verzicht auf mandantenabhängige Funktionsausführung (S)

ABAP-Programme SOLLTEN NICHT mandantenabhängig programmiert werden, so dass sie nur von einem bestimmten Mandanten ausgeführt werden können. Sollte dies jedoch unbedingt erforderlich sein, SOLLTE es detailliert dokumentiert werden. Außerdem SOLLTEN dann zusätzliche Sicherheitsmaßnahmen ergriffen werden, wie beispielsweise eine manuelle Code-Überprüfung (manuelles Code-Review) oder eine Qualitätssicherung auf dem entsprechenden Mandanten.

APP.4.6.A18 Vermeidung von Open-SQL-Injection-Schwachstellen (S)

Dynamisches Open SQL SOLLTE NICHT verwendet werden. Falls Datenbankzugriffe mit dynamischen SQL-Bedingungen notwendig sind, SOLLTEN KEINE Eingaben von Benutzenden in der jeweiligen Abfrage übertragen werden. Wenn das dennoch der Fall ist, SOLLTEN die Eingaben von Benutzenden zwingend geprüft werden (Output Encoding).

APP.4.6.A19 Schutz vor Cross-Site-Scripting (S)

Auf selbst entwickeltes HTML in Business-Server-Pages-(*BSP*)-Anwendungen oder HTTP-Handlern SOLLTE möglichst verzichtet werden.

APP.4.6.A20 Keine Zugriffe auf Daten eines anderen Mandanten (S)

Die automatische Mandantentrennung SOLLTE NICHT umgangen werden. Auf Daten anderer Mandanten SOLLTE NICHT mittels *EXEC SQL* oder der Open SQL Option *CLIENT SPECIFIED* zugegriffen werden.

APP.4.6.A21 Verbot von verstecktem ABAP-Quelltext (S)

Der Quelltext eines selbst erstellten ABAP-Programms SOLLTE immer lesbar sein. Techniken, die das verhindern (Obfuskation), SOLLTEN NICHT verwendet werden.

3.3. Anforderungen bei erhöhtem Schutzbedarf

Im Folgenden sind für diesen Baustein exemplarische Vorschläge für Anforderungen aufgeführt, die über dasjenige Schutzniveau hinausgehen, das dem Stand der Technik entspricht. Die Vorschläge SOLLTEN bei erhöhtem Schutzbedarf in Betracht gezogen werden. Die konkrete Festlegung erfolgt im Rahmen einer individuellen Risikoanalyse.

APP.4.6.A22 Einsatz von ABAP-Codeanalyse Werkzeugen (H)

Zur automatisierten Überprüfung von ABAP-Code auf sicherheitsrelevante Programmierfehler, funktionale und technische Fehler sowie auf qualitative Schwachstellen SOLLTE ein ABAP-Codeanalyse-Werkzeug eingesetzt werden.

4. Weiterführende Informationen

4.1. Wissenswertes

Im „Best Practice Guide: Leitfaden Development ABAP 2.0“ der Deutschsprachigen SAP Anwendergruppe e.V. (DSAG) finden sich vertiefende Informationen zur ABAP-Programmierung.

Weitere Informationen und Best Practices zur sicheren ABAP-Programmierung finden sich im Buch „Sichere ABAP-Programmierung“ von Wiegenstein, Schuhmacher, Schinzel, Weidemann aus dem SAP Press Verlag.