Bundesamt
für Sicherheit in der
Informationstechnik

EMPFEHLUNG: IT-HERSTELLER

Entwicklung sicherer Webanwendungen

Empfehlungen für einen sicheren Lebenszyklus

Während in der Vergangenheit Anwendungen meist lokal auf dem PC der Anwender betrieben wurden, hat sich der Trend zu webbasierter Software für sämtliche Einsatzgebiete verstärkt. Dies ist einer der Gründe dafür, dass Webseiten schon seit geraumer Zeit E-Mails als primären Verbreitungspfad für Schadsoftware abgelöst haben. Zudem waren in der jüngeren Vergangenheit zunehmend Kompromittierungen von Webanwendungen zu verzeichnen, die den Verlust bzw. Abfluss von – häufig unternehmenskritischen und/oder personenbezogenen – Daten zur Folge hatten.

Dieses Dokument gibt einen Überblick über mögliche Vorgaben an den Auftragnehmer eines Projekts zur Entwicklung und Bereitstellung sicherer Webanwendungen. Die Empfehlungen orientieren sich am Lebenszyklus einer Webanwendung, welcher typischerweise u. a. die folgenden Abschnitte beinhaltet, die in den gängigen Vorgehensmodellen zur Anwendungsentwicklung so oder ähnlich vorzufinden sind:

1. Vergabephase
2. Planungs- und Konzeptionsphase
3. Fehler: Referenz nicht gefunden
4. Test- und Rollout-Phase
5. Betriebsphase
6. Ende des Lebenszyklus

Diese Vorgaben können beispielsweise im Rahmen einer Ausschreibung den Verdingungsunterlagen beigestellt werden. Zuvor sollte der Auftraggeber diese jedoch einer Prüfung unterziehen und ggf. Ergänzungen oder Konkretisierungen (insbes. hinsichtlich Zeiten, Fristen, etc.) vornehmen und nicht relevante Passagen streichen. Letzteres gilt insbesondere, falls lediglich die Entwicklung, aber nicht der Betrieb einer Webanwendung beauftragt wird.

Durch solche einheitlichen, fundierten Vorgaben für diese Abschnitte im Lebenszyklus einer Webanwendung kann das Niveau der IT-Sicherheit deutlich verbessert und ein wichtiger Schritt zum Schutz vor Cyber-Angriffen beigetragen werden. Parallel ergeben sich automatisch Anwendungsmöglichkeiten für die Beschaffung von Produkten (Commercial-off-the-shelf / COTS statt Individualentwicklung) und die Entwicklung von Nicht-Webanwendungen, da ein Großteil der Aspekte und Kriterien dort ebenfalls anwendbar ist.

1 Vergabephase

Insbesondere bei Webanwendungen, die von externen Auftragnehmern entwickelt werden, haben zentrale Vorgaben bezüglich Personal, Prozessen und vertraglichen Regelungen einen besonderen Stellenwert, um die Sicherheit des späteren Produkts zu gewährleisten. Dazu gehören insbesondere:

- Erfüllung der sicherheitsspezifischen Vorgaben: Sämtliche in der Leistungsbeschreibung dargestellten sicherheitsspezifischen Anforderungen und Rahmenbedingungen müssen zwingend vom Angebot erfüllt werden. Dies können z. B. Compliance-Anforderungen im Kontext von PCI DSS[1], das Bundesdatenschutzgesetz (BDSG), das Jugendschutzgesetz oder die vom Auftraggeber vorgesehenen Nutzungsbedingungen für die Webanwendung sein. Weicht ein Bieter im Angebot von den sicherheitsspezifischen Vorgaben (z. B. wegen unvertretbar hohem Aufwand) ab, muss dies explizit dokumentiert und begründet werden.
- Eingesetztes Personal: Während der gesamten Projektlaufzeit kommt ausschließlich genau das Personal zum Einsatz, welches im Angebot genannt wurde und für welches die Qualifikation hinreichend belegt wurde. Ergänzend werden genaue Angaben zu Konsortien, Unterbeauftragungen sowie Near- und Offshore-Modellen gemacht. Die Auslagerung von Arbeitspaketen an Subunternehmer bedarf der expliziten schriftlichen Zustimmung. An Subunternehmer oder Unterbeauftragte sind mindestens die gleichen Sicherheitsanforderungen zu stellen, die auch seitens des Auftraggebers für den Auftragnehmer bzw. Konsortialführer selbst definiert wurden.
- Feste Ansprechpartner: Es erfolgt die verbindliche Nennung eines Gesamtverantwortlichen für das Projekt sowie eines Ansprechpartners für Sicherheitsaspekte inkl. jeweils eines Stellvertreters mit garantierten Erreichbarkeiten für den gesamten Lebenszyklus der Webanwendung.
- Organisatorische Aspekte: Mit dem Angebot legt der Bieter die von ihm umgesetzten Prozesse mit sicherheitsspezifischer Relevanz dar. Hierzu gehören u. a. Vorgehensmodelle der Softwareentwicklung, Freigabe- und Testverfahren sowie Qualitätssicherungsprozesse und die dabei angewandten Standards. Bei Tests ist das Vier-Augen-Prinzip – also die Trennung der Zuständigkeiten für Entwicklung und Tests – essenziell. Darüber hinaus sind die etablierten physikalischen, organisatorischen und personellen Sicherheitsmaßnahmen darzustellen, die zur Gewährleistung einer sicheren Entwicklungsumgebung dienen. Bei Bedarf kann die Effektivität dieser Maßnahmen durch ein externes Audit verifiziert werden.
- Best Practices: Der Anbieter fügt dem Angebot eine Beschreibung bzw. eine Übersicht der Best Practices bei, die er standardmäßig in Softwareprojekten zur Gewährleistung der IT-Sicherheit anwendet. Neben Programmierrichtlinien umfasst dies z. B. Compileroptionen und Bibliotheken.
- Sichere Entwicklungs-, Test- und Staging-Systeme: Sämtliche Systeme für Testing und Staging müssen gemäß der Vorgaben für den Produktivbetrieb gehärtet und konfiguriert sein. Es dürfen allerdings keinesfalls dieselben Systeme für die unterschiedlichen Zwecke verwendet werden. Die Entwicklungsumgebung ist gemäß aktueller Best Practices ebenfalls gegen Cyber-Angriffe abzusichern.
- Service Level Agreements: Insbesondere bei der Übernahme von Betriebsaufgaben sind die jeweiligen Rahmenbedingungen für abzuschließende Service Level Agreements (SLA) durch den Bieter mit Abgabe des Angebots zu bestätigen.
- Behandlung von Sicherheitslücken und -vorfällen: Im Falle des Bekanntwerdens von Sicherheitslücken bzw. beim Auftreten von Sicherheitsvorfällen garantiert der Bieter die Einhaltung der geforderten Reaktionszeiten hinsichtlich Überprüfung und Klassifizierung des Sachverhalts, Information des Kunden, Lieferung von Workarounds und der Behebung der Schwachstelle. Die in einem solchen Fall anzuwendenden Prozesse sind detailliert darzustellen und müssen verpflichtend umgesetzt werden. Darüber hinaus ist das Prozedere zu beschreiben, mithilfe dessen Schwachstellen von Kunden oder Externen an den Auftragnehmer gemeldet werden können. Der Anbieter verpflichtet sich zur zeitnahen Information des Auftraggebers hinsichtlich potenzieller oder verifizierter Schwachstellen – unabhängig davon, ob bereits ein Patch existiert oder nicht. Die Freigabe von Updates bzw. Patches folgt einem definierten und dokumentierten Prozess.
- Produktverfügbarkeit, Support und Patches: Der Auftragnehmer gewährleistet die Verfügbarkeit sämtlicher verwendeten Komponenten für den gesamten Lebenszyklus der Webanwendung. Zudem wird die Bereitstellung von Sicherheitspatches nach vorherigen Tests sowie Support für sämtliche

1 Payment Card Industry Data Security Standard, https://www.pcisecuritystandards.org/

Komponenten über die gesamte Projektlaufzeit gewährleistet. Dies gilt explizit auch für Komponenten anderer Hersteller (Web-/Application-/Portalserver, Datenbank, Bibliotheken, Betriebssystem, etc.). Evtl. anzuwendende Fristen werden vertraglich geregelt.

- Hinterlegung des Sourcecodes: Der Sourcecode wird für die gesamte Projektlaufzeit sowie darüber hinaus für einen festzuschreibenden Zeitraum bei einer vertrauenswürdigen Stelle (z. B. Notar) hinterlegt, damit beispielsweise im Falle einer Insolvenz des Auftragnehmers Updates für kritische Schwachstellen erstellt werden können.
- Vertraulichkeit: Der Auftragnehmer gewährleistet die sichere Datenhaltung und Übertragung für die gesamte Projektlaufzeit. Die Rückgabe, Vernichtung oder der Verbleib von Dokumenten nach Projektabschluss wird vertraglich geregelt.
- Wartungsaktivitäten: Die Vorgaben des Auftraggebers für Fern- und Vor-Ort-Wartung durch Hersteller bzw. beauftragte Dienstleister werden beachtet.
- Konfigurations- und Change-Management: Der Auftragnehmer betreibt ein systematisches Konfigurations- und Change-Management zur Gewährleistung der zeitnahen Umsetzung erforderlicher Änderungen sowie zur Planbarkeit und dem Investitionsschutz seitens des Auftraggebers.

2 Planungs- und Konzeptionsphase

Aufgaben / Deliverables:

- Sicherheitskonzept: Die Erarbeitung eines ganzheitlichen Sicherheitskonzepts zur Umsetzung der erforderlichen Sicherheitsmechanismen umfasst u. a. die Schritte *Strukturanalyse*, *Schutzbedarfsfeststellung* und *Risikoanalyse*. Eine Grundlage hierzu liefert der IT-Grundschutz[2].
- Grobkonzept: Das Grobkonzept liefert eine Übersicht über die Architektur der Webanwendung und das Gesamtsystem inklusive sämtlicher Drittkomponenten. Die Dokumentation sollte alle Bestandteile der Web-Anwendung berücksichtigen. Dabei sollten mindestens alle Abhängigkeiten (z. B. zu Frameworks, Bibliotheken, Betriebssystemen oder Hardware), Schnittstellen (z. B. zu Hintergrundsystemen) und Interaktionen sowie für den Betrieb notwendige Drittkomponenten der Web-Anwendung dokumentiert werden. Darüber hinaus werden sämtliche sicherheitsspezifische Entwurfsentscheidungen und Mechanismen dargestellt. Insgesamt soll dieses Dokument eine Bewertung des erzielten Sicherheitsniveaus ermöglichen. Aus der Dokumentation sollte hervorgehen, welche Komponente welche Sicherheitsmechanismen umsetzt (z. B. Benutzermanagement, Authentisierung, Autorisierung, Session-Management, Protokollierung oder Transportsicherheit).
- Feinkonzept: Das Feinkonzept vertieft die Darstellungen des Grobkonzepts und stellt die jeweiligen Details dar. Hierzu gehören insbesondere Ausführungen zu sicherheitsrelevanten Systemkomponenten und Implementierungsdetails bzw. Design-Entscheidungen mit Bezug auf das Sicherheitskonzept.
- Testkonzept: Für sämtliche Arten von Tests und Reviews sind geeignete Testpläne und –prozeduren bereits in der Planungsphase zu erstellen und anschließend fortzuschreiben. Diese sind als Teil der Projektergebnisse an den Auftraggeber zu übergeben. Darüber hinaus erfolgt ein nachvollziehbarer Soll-Ist-Vergleich sämtlicher Testergebnisse.

Die Fortschreibung dieser Dokumente erfolgt während des gesamten Projektverlaufs.

Für die Phase der Planung gelten allgemeine sicherheitsspezifische Anforderungen, deren Einhaltung in den o. g. Dokumenten nachvollziehbar belegt sein muss.

- Sicherheitsspezifische Grundprinzipien: Im gesamten Planungsprozess sind anerkannte Grundregeln und -prinzipien zu berücksichtigen, wie z. B. Defense in depth, Multiple Independent Levels of Security (MILS), sichere Fehlerzustände oder das Minimalitätsprinzip für Rollen und Berechtigungen.
- Modulare Anwendungsarchitektur: Die Architektur ist geprägt durch einen modularen Aufbau einzelner, getrennter Komponenten und Schichten (z. B. Präsentations-, Anwendungs- und Datenschicht). Diese werden zunächst jeweils separat möglichst sicher ausgestaltet. Anschließend

2 BSI IT-Grundschutz: https://www.bsi.bund.de/DE/Themen/ITGrundschutz/itgrundschutz_node.html

erfolgt die Betrachtung und ergänzende Absicherung der Gesamtarchitektur. Hierdurch wird eine bestmögliche Sicherheit auch bei der Wiederverwendung von Teilkomponenten in einem anderen Kontext gewährleistet.

- Definierte Schnittstellen: Die Schnittstellen zwischen den Teilsystemen werden sauber definiert und dokumentiert. Backdoors oder alternative Zugriffsmöglichkeiten auf Teil- oder Hintergrundsysteme sind nicht zulässig. Es erfolgt eine geeignete Filterung von Ein- und Ausgabedaten zwischen den Teilsystemen, um die Auswirkungen von Angriffen zu beschränken.
- Komponentenauswahl: Die Auswahl sicherer Basiskomponenten, Laufzeitumgebungen, Bibliotheken, etc. ist unter dem Aspekt der Sicherheit sorgsam zu treffen und zu dokumentieren. Abhängigkeiten (Lock-In) bzgl. einzelner Hersteller, Technologien oder Versionsstände sind zu vermeiden.
- Patchfähigkeit: Die Webanwendung muss möglichst einfach mit Sicherheitsupdates zu versorgen sein.

Darüber hinaus werden in der Planungsphase Festlegungen in der Gestalt von Sicherheitsvorgaben und Implementierungsentscheidungen getroffen, um die im Sicherheitskonzept dargestellten Sicherheitsziele auszugestalten. Eine Übersicht der wichtigsten Aspekte befindet sich im folgenden Abschnitt.

3 Entwicklungsphase

Für die Phase der Entwicklung gelten die folgenden Vorgaben zur Implementierung von Sicherheitsmechanismen für die Webanwendungen sowie ggf. weiterer Schnittstellen (z. B. API):

- Validierung der Eingabedaten: Zwingend umzusetzen ist die kontextsensitive Validierung und Plausibilitätsprüfung sämtlicher Ein- und Ausgabedaten nach Kanonisierung bzw. Dekodierung durch Escaping von Metazeichen, Blacklisting von Keywords oder Whitelisting im jeweiligen Kontext (HTML, JS, CSS, URL). Darüber hinaus müssen Ein- oder Ausgabedaten hinsichtlich Wertebereich/Zeichen, Typ/Format und Länge geprüft werden.
  Die Validierung von Benutzereingaben ist auch für Parameter relevant, die an externe durch die Webanwendung aufgerufene Programme weitergegeben werden, um Command-Injection zu vermeiden.
- Validierung der Ausgaben: Auch Ausgaben der Webanwendung, wie zurückgespielte Nutzereingaben oder Einträge aus Datenbanken, müssen validiert werden.
- Verhinderung von Injection Angriffen: Eingabedaten und Befehle sind strikt voneinander zu trennen: Soweit möglich ist eine sichere API ohne Aufruf von Interpretern oder mit typgebundenen Schnittstellen zu verwenden. Darüber hinaus sollten keine String-Konstrukte als Eingabe für Drittsysteme (z. B. SQL-Befehle oder LDAP-Anfragen) verwendet werden. Prinzipiell sind Stored Procedures am besten geeignet, um sichere Abfragen von Datenbanken vorzunehmen. Weitere Möglichkeiten sind Prepared Statements oder object-relational mapping (z. B. Hibernate). Bei allen diesen Umsetzungsmöglichkeiten muss allerdings sichergestellt werden, dass keine Manipulationen über Nutzereingaben oder sonstige Schnittstellen erfolgen kann.
- Verhinderung von Cross-Site Request Forgery (CSRF): Es ist ein nicht-vorhersagbares Token pro Request oder zumindest pro Sitzung zu verwenden. Ein solches Token sollte im Body des HTTP-Request in Form eines Hidden Input Field übertragen werden. Die Übertragung von Token (sowie anderer kritischer Daten) als URL-Parameter ist zu vermeiden.
- Authentisierung und Autorisierung: Es werden lediglich bewährte bzw. geprüfte Verfahren und Implementierungen eingesetzt. Rollen- und Policy-basierte Ansätze sind zu bevorzugen. Bei Passwort-basierter Authentisierung sollte eine Passwortrichtlinie etabliert und auch auf technischer Ebene forciert werden, welche die Qualität von Passwörtern sowie deren periodische Änderung u. ä. regelt. Die Prüfung der Autorisierung muss bei jedem Aufruf einer Ressource erfolgen. Dabei ist ein Whitelisting-Ansatz zu bevorzugen, d. h. es werden sämtliche Zugriffe verboten, die nicht explizit erlaubt wurden (default-deny).
- Session Management: Es werden Session-Frameworks für eine sichere Handhabung der Sitzungsdaten eingesetzt. Session IDs werden ausschließlich als Cookie im HTTP-Header übergeben. Der Austausch der Sitzungsdaten erfolgt über sichere Kanäle. Die Erzeugung der Session

IDs erfolgt unter Verwendung kryptografischer Zufallszahlengeneratoren. Bei sicherheitskritischen Anfragen sind geeignete Token zu verwenden, um Session Riding zu verhindern. Sessions werden mit einem Time-out versehen. Benutzer können eine Session auch manuell beenden. In beiden Fällen werden die Sitzungsdaten zerstört. Eine Session sollte möglichst an die jeweilige IP-Adresse gebunden sein.

- Plausibilitätsprüfung: Durch die Analyse sämtlicher Aufrufe und Parameter werden insbesondere solche Aufrufe innerhalb einer Webanwendung detektiert, die nicht im Kontext einer ordnungsgemäßen Sitzung bzw. eines Workflows stehen.
- Referenzen auf Objekte/Ressourcen: Objektreferenzen sind möglichst indirekt zu realisieren. Lässt sich eine direkte Objektreferenzierung nicht vermeiden, so müssen bei jedem Aufruf die Zugriffsberechtigungen geprüft werden.
- Um- und Weiterleitungen: Um- und Weiterleitungen sollten möglichst vermieden werden. Insbesondere sind keine Um- oder Weiterleitungen in Abhängigkeit von Nutzereingaben erlaubt. Bei jeder Um- oder Weiterleitung muss eine Prüfung der Zugriffsberechtigungen erfolgen.
- Verhinderung von Buffer Overflows: Es sind geeignete Maßnahmen zu treffen, um Buffer Overflows zu vermeiden. Dies kann neben dem Verzicht auf anfällige Plattformen und kritische Mechanismen (z. B. native Aufrufe) durch eine hinreichende Filterung umgesetzt werden.
- Sicherer Einsatz bzw. Umsetzung von kryptografischen Mechanismen: Es erfolgt eine sorgfältige Auswahl hinreichend sicherer Algorithmen, Parameter und Schlüssellängen, wobei insbesondere der Katalog der BNetzA[3] beachtet wird. Kryptografische Mechanismen werden ausschließlich auf der Basis etablierter und bewährter Implementierungen umgesetzt. Die Verwendung von Eigenimplementierungen ist nicht zulässig. Zudem ist besondere Sorgfalt hinsichtlich der Verwaltung und Speicherung von kryptografischen Schlüsseln geboten. Die Verwendung sicherheitsspezifischer Protokolle und Technologien (z. B. SSL oder SAML) erfolgt ausschließlich in sicheren Varianten.
- Sichere Speicherung und Übertragung schutzbedürftiger Daten: Schutzbedürftige Daten dürfen nicht im Klartext gespeichert oder übertragen werden. Bei Passwörtern empfiehlt sich die Speicherung als Salted Hash unter Verwendung hinreichend sicherer Hashfunktionen.
- Konfigurationsdateien: Die Konfiguration der Anwendung selbst sowie die Parametrierung von verwendeten Bibliotheken muss möglichst sicher gestaltet werden. Konfigurationsdateien müssen vor unberechtigtem Zugriff geschützt sein (Zugriffsrechte, Verschlüsselung) und dürfen nicht über die Anwendung ausgelesen/geändert werden können.
- Error Handling: Sämtliche Fehler (Exceptions) sind sicher zu handhaben. System-und Fehlermeldungen dürfen keine sicherheitskritischen Informationen über das System enthalten. Es sind möglichst einheitliche Fehlermeldungen zu verwenden, die einem Angreifer keine Rückschlüsse auf interne Verarbeitungsfehler (z. B. bei kryptografischen Operationen) liefern.
- Logging: Es wird ein Konzept für das Logging innerhalb der Webanwendung erarbeitet. Ein zentrales Logging außerhalb der Webanwendung sowie in unterschiedlichen Datensenken ist zu ermöglichen. Mindestens sämtliche sicherheitsrelevanten Ereignisse sind dem Logging-Mechanismus zuzuführen. Die Detailstufe des Logging muss im laufenden Betrieb flexibel angepasst werden können. Logdaten sollten vor Manipulationen geschützt sein oder zumindest Manipulationsversuche erkennbar machen (z. B. mittels kryptografischer Timestamps).
- Debugging: Debugging-Mechanismen sind lediglich in der Phase der Entwicklung zu verwenden und anschließend zu deaktivieren. Angefallene Debugging-Informationen sind zu löschen.
- Nutzung sicherer Verbindungen: Zumindest für sämtliche Seiten mit schutzbedürftigen Inhalten muss SSL verwendet werden. Zugriffe über http sind entsprechend auf https umzuleiten. Das Secure Flag ist für sämtliche kritischen Cookies zu setzen. Gleichermaßen sind die Zugangsmöglichkeiten für Administratoren mittels Weboberfläche oder anderer Zugänge (z. B. scp statt ftp) abzusichern.
- Analyse des Nutzerverhaltens: Auf Mechanismen zur Analyse des Nutzerverhaltens bzw. zur Identifizierung der Nutzer ohne deren explizite Zustimmung wird verzichtet. Insbesondere auf sogenanntes Web Tracking mit Hilfe von Cookies, Tracking-Pixeln oder Browser Fingerprints ist zu verzichten. Externe Dienste zum Web Tracking sind prinzipiell nicht zu verwenden.

3 https://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/QES/Veroeffentlichungen/Algorithmen/2016Algorithmenkatalog.html

- Abwehr von Brute-Force-Angriffen: Die Webanwendung verfügt über die Fähigkeit zur Erkennung von Brute-Force-Angriffen. Darüber hinaus ist eine Tarpit-Funktionalität (Teergrube) mit sinnvollem Timing und Maßnahmen gegen Verschleierungstaktiken zu implementieren.
- Vermeiden trügerischer Sicherheit: Auf Mechanismen, die nur scheinbar ein Plus an Sicherheit liefern oder sogar zur Schwachstelle werden können, wird verzichtet. Hierzu gehören beispielsweise „Passwort-Vergessen-Funktionen“ oder Captchas, sofern diese nicht den Vorgaben des Auftraggebers entsprechen.
- Sicherer Umgang mit Dateien und Einbinden von Dateien: Das Einbinden von Dateien bzw. das Annehmen und Erzeugen von Dateien wird auf das Notwendigste eingeschränkt. Eingebundene Dateien sowie deren Pfade dürfen nicht durch Nutzer manipuliert werden können. Das Einbinden von Dateien über das Netz wird deaktiviert oder über eine Whitelist abgesichert.
- Upload und Erzeugung von Dateien: Ordner- und Dateinamen für hochgeladene Dateien werden zufällig und nicht-vorhersagbar gewählt. Die Anwender können den Pfad zur Speicherung von Dateien nicht modifizieren. Der Inhalt der Dateien wird einer Plausibilitäts- und Virenprüfung unterzogen. Uploadmechanismen sind durch konfigurierbare Beschränkungen hinsichtlich Größe, Dateityp und Anzahl limitiert. Anwender können hochgeladene bzw. erzeugte Dateien nicht auf dem Webserver ausführen. Bilder und Dokumente sollten konvertiert werden, um Schadcode zu entfernen und Fehlformatierungen zu korrigieren.
- Nicht benötigte Dateien: Nicht benötigte Dateien im Wurzelverzeichnis des Servers werden entfernt. Dateien mit Programmcode werden in jedem Fall vom Server interpretiert und nicht im Quellcode ausgeliefert.
- Schutz vor Massenabfragen: Jegliche Art von Massenabfragen auf Nutzerdaten ist zu verhindern (z. B. Crawler-Zugriffe und API).
- Sicherheit von Links: Der Verschleierung von URLs ist entgegenzuwirken. Wird ein Link (URL) durch einen Nutzer in der Webanwendung veröffentlicht bzw. eingefügt, sollte dieser komplett – inklusive aller Parameter – sichtbar sein. Kurz-URLs sind aufzulösen.
- Sicherheitsspezifische Online-Hilfefunktionen: Es wird eine kontextsensitive Hilfe zu sicherheitsspezifischen Funktionen und Einstellungen implementiert. Darüber hinaus werden Kontaktmöglichkeiten für (vermutete) Sicherheitsvorfälle genannt.
- Barrierefreiheit: Die Webanwendung ist barrierefrei gemäß Behindertengleichstellungsgesetz des Bundes (BGG) und der Verordnung zur Barrierefreien Informationstechnik (BITV) zu realisieren. Bei der Verwendung von Aktiven Inhalten sollte darauf geachtet werden, dass sich die Webanwendung auch bei deaktivierten Aktiven Inhalten nutzen lässt.
- Anbindung an Hintergrundsysteme: Beim Zugriff auf Hintergrundsysteme, beispielsweise Datenbanken, ist eine Authentisierung erforderlich. Ein Zugriff sollte immer über wohldefinierte Schnittstellen, wie Stored Procedures, Prepared Statements oder Frameworks erfolgen. Die Kommunikation zwischen Webanwendung und Hintergrundsystemen erfolgt i.d.R. verschlüsselt.

Diese allgemeinen Vorgaben müssen um Policies oder Best Practices für die verwendete Programmiersprache bzw. für das verwendete Web Application Framework ergänzt werden.

Durch fortlaufende Reviews zur statischen und ggf. dynamischen Codeanalyse wird die Einhaltung der Vorgaben während der Entwicklung sowie im Zuge der Abnahme überprüft. Insbesondere statische Codeanalysen sollten bereits im Rahmen der Entwicklungsphase begleitend durchgeführt werden, beispielsweise auf der Basis von Nightly Builds. Im Fokus stehen dabei Schwachstellen sowie schlechter Programmierstil (z. B. Dead Code). Es dürfen nach Fertigstellung zudem keine Daten, Ressourcen oder Funktionen aus der Entwicklungs- oder Testphase enthalten sein.

4 Test- und Rollout-Phase

Für die Auslieferung bzw. Übergabe der Webanwendung gelten die folgenden Anforderungen:

- Softwaretests: Vor Auslieferung wird die entwickelte Software in der vereinbarten Standardkonfiguration seitens des Auftragnehmers durch ein Team getestet, welches nicht identisch mit dem Entwicklerteam ist (Vier-Augen-Prinzip).

- Penetrationstests: Während der Code-Review auf der Basis sämtlicher Informationen inkl. Quellcode durchgeführt wird, erfolgt der Penetrationstest typischerweise aus der Perspektive eines (externen) Angreifers. Dabei wird versucht, Schwachstellen in der Web-Anwendung zu identifizieren und diese auszunutzen, um die identifizierten Schutzziele zu verletzen. Die Testfälle in einem Penetrationstest beziehen sich typischerweise auf Aspekte wie Eingabevalidierung, Session Management, Konfigurationsmanagement oder Authentisierung.
- Sichere Standardkonfiguration: Nach der Installation befindet sich die Webanwendung in einer sicheren Konfiguration. Ggf. erforderliche Maßnahmen (z. B. Ändern von Standardpasswörtern) zur Erreichung einer betriebssicheren Konfiguration werden übersichtlich und deutlich dokumentiert und falls möglich auch technisch unterstützt bzw. forciert.
- Vertraulichkeit und Integrität der Auslieferung: Es wird eine Möglichkeit zum sicheren Download sowie zur Integritätsprüfung der ausgelieferten Software (Sourcen und Binaries) bereitgestellt (z. B. Prüfsummen). Zusätzlich werden Codesignaturen angebracht, um die Integrität der Webanwendungen zu gewährleisten. Die verwendeten Software-Repositories sind mit hinreichenden Sicherheitsmechanismen zu versehen.
- Dokumentation für Anwender: Die (Online-)Dokumentation für die Anwender enthält die Beschreibung aller sicherheitsrelevanten Einstellungen und deren Defaultwerte sowie alle Systemmeldungen und jeweils infrage kommende Ursachen und Behebungsmaßnahmen. Sämtliche sicherheitsspezifischen Funktionen werden benannt. Darüber hinaus werden die wichtigsten Grundregeln für die sichere Verwendung angegeben.
- Dokumentation für Administratoren: Es wird eine umfassende Dokumentation der Voraussetzungen und Umgebungsanforderungen (Netzwerk, Infrastruktur, Web-/Application-/Portal-/CMS-Server, Betriebssystem, Datenbanken, flankierende Sicherheitsmechanismen, etc.) für den sicheren Betrieb angefertigt. Diese enthält die Beschreibung aller sicherheitsrelevanten Einstellungen und deren Defaultwerte sowie alle Systemmeldungen und jeweils infrage kommende Ursachen und Behebungsmaßnahmen. Sämtliche sicherheitsspezifischen Funktionen werden benannt. Darüber hinaus werden die wichtigsten Grundregeln für den sicheren Betrieb genannt. Bei Konfigurationsmöglichkeiten mit potenziell sicherheitskritischen Folgen wird auf die möglichen Konsequenzen hingewiesen. Es wird auf sicherheitsrelevante Aspekte hingewiesen, die im Rahmen der Wartung und Pflege zu beachten sind.

5 Betriebsphase

Abschließend wird die entwickelte Softwarekomponente in den Betrieb überführt. Auch hier ist die Einhaltung verschiedener Vorgaben erforderlich, um ein hinreichendes Maß an Sicherheit zu gewährleisten. Viele dieser Maßnahmen erfordern lediglich einen einmaligen Aufwand sowie eine regelmäßige Prüfung und Pflege, die allerdings mitunter nur geringe Ressourcen in Anspruch nimmt.

- Security Management: Die Webanwendung wird in bestehende GRC-Prozesse (Governance, Risk & Compliance) sowie Konzepte für Notfallmanagement, Datensicherung, Asset Management, Versionierungs- & Patch Management integriert. Von besonderer Bedeutung ist dabei die Festlegung von Rollen und Verantwortlichkeiten für den Betrieb.
- Verwendung einer Web Application Firewall (WAF): Es werden geeignete WAFs zur Prüfung und Filterung von Metazeichen und kritischen Inhalten nach Kanonisierung der Eingabedaten eingesetzt. Darüber hinaus sollte eine WAF gewählt werden, die Brute Force- und (D)DoS-Angriffe detektieren kann.
- Härtung der Serversysteme: Durch die restriktive Konfiguration der Serversysteme nach dem Minimalitätsprinzip werden mögliche Angriffspfade reduziert. Hierzu gehört die Beschränkung der auf einem Server laufenden Dienste auf das absolut Notwendige sowie die Schärfung der Berechtigungen im Dateisystem, insbesondere bzgl. der Webserver. Darüber hinaus sind die Serverdienste mit den minimal erforderlichen Rechten zu betreiben. Standardmäßig konfigurierte User/Accounts sollten deaktiviert oder gelöscht bzw. Passwörter geändert werden. Zudem sollten nicht benötigte Produktfunktionen (z. B. von CMS-Systemen oder Portalservern) deaktiviert werden. Der Härtungsprozess sollte möglichst automatisiert und reproduzierbar realisiert werden. Es sollten identisch gehärtete Systeme für Tests und Produktivbetrieb verwendet werden.

- Absicherung des Webservers: Der Webserver sollte in einer Chroot-Umgebung betrieben werden, um zu verhindern, dass ein Angreifer aus einem kompromittierten Webserver ausbrechen und die Kontrolle über weitere Komponenten übernehmen kann. Für einige Webserver sind spezielle Sicherheits-Frameworks verfügbar, die zur Absicherung beitragen können (z. B. mod_security für Apache).
- Transportsicherheit durch SSL: Die Verwendung hinreichend sicherer Algorithmen bzw. Parameter für SSL-Verbindungen ist sicherzustellen. Es sind gültige Zertifikate zu verwenden, deren Root im Browser (bzw. Betriebssystem, JRE, etc.) enthalten ist. Auch Verbindungen im Backend sind mit SSL oder durch andere Tunnel zu schützen.
- Fernwartung: Fernwartungszugriffe seitens Hersteller/Dienstleister müssen abgesichert, minimiert sowie personalisiert werden und dürfen erst nach Freigabe durch den Betreiber erfolgen. Der Zugang muss immer über eine Firewall geführt werden. Zugriffe bzw. Zugriffsversuche werden durch einen Logmechanismus erfasst.
- Sensibilisierung der Nutzer: Anwender als schwächstes Glied der Sicherheitskette sind durch geeignete Sicherheitshinweise (z. B. zur Verwendung sicherer Passwörter, dem Einsatz aktueller Browser oder der Verwendung aktueller Antiviren-Lösungen) zu sensibilisieren.
- Einspielen von Patches: Updates und Patches für Betriebssysteme, Web-/Application-/Portalserver, Datenbanken, Bibliotheken sowie weiterer Dritt- und Sicherheitskomponenten sollten zeitnah eingespielt werden. Insbesondere Erweiterungen von Webanwendungen (Plug-ins, Add-ons) sowie Bibliotheken sollten nur aus vertrauenswürdigen Quellen und nach vorheriger Virenprüfung und Testphase eingesetzt werden.
- Schwachstellenanalysen: Die regelmäßige Durchführung von Audits, Schwachstellenscans und Penetrationstests ermöglicht das Beheben von Schwachstellen, bevor diese durch Angreifer ausgenutzt werden können. Sie sind daher verpflichtend und regelmäßig durchzuführen.
- Auswertung von Logdateien: Logdateien werden automatisiert ausgewertet. Dabei wird insbesondere nach Anzeichen für versuchte oder auch erfolgreich durchgeführte Angriffe gesucht. Ergänzend erfolgt eine regelmäßige manuelle Prüfung der Logdaten.
- Backup- & Recovery-Strategie: Die Datensicherung von Konfigurations- und Nutzdaten (insbesondere Datenbanken und Verzeichnisdienste) erfolgt regelmäßig und unter Nutzung kryptografischer Schutzmechanismen zur Gewährleistung der Vertraulichkeit und Integrität.

6 Ende des Lebenszyklus

Wird der Betrieb einer Webanwendung eingestellt, sind abschließende Anforderungen zu erfüllen, wie z. B.:

- Umsetzung von Regelungen zum Verbleib von Daten, Löschpflichten und -fristen
- Widerruf / Sperrung von Zertifikaten

7 Weiterführende Informationen

Ergänzend zu den serverseitigen Maßnahmen müssen insbesondere die Clientsysteme der Administratoren hinreichend abgesichert werden. Hierzu sind vor allem die Maßnahmen der BSI Empfehlung „PCs unter Microsoft Windows - für kleine Unternehmen und Selbstständige" bzw. vergleichbare Maßnahmen für andere Betriebssysteme umzusetzen.

Weiterführende Informationen zur Absicherung von Webanwendungen und deren Infrastruktur (IT-Systeme, Netze, etc.) liefert der BSI-Standard zur Internet-Sicherheit (ISi-Reihe[4]). Diese enthält Studien für IT-Fachleute, Leitlinien für IT-Führungskräfte sowie Checklisten für Administratoren, Programmierer und Web-Entwickler.

Mit den BSI-Veröffentlichungen publiziert das Bundesamt für Sicherheit in der Informationstechnik (BSI) Dokumente zu aktuellen Themen der Cyber-Sicherheit. Kommentare und Hinweise können von Lesern an info@cyber-allianz.de gesendet werden.

4 https://www.bsi.bund.de/ISi-Reihe/