Qualitätsicherung Care (PpUGV)

|  |  |
| --- | --- |
| **Autor** | **Revisor** |
| Müller, Michael |  |
|  |  |
|  |  |

# Überblick

Entsprechend der Verordnung zur Festlegung von Pflegepersonaluntergrenzen in pflegesensitiven Krankenhausbereichen für das Jahr 2019 (Pflegepersonaluntergrenzen-Verordnung - PpUGV) hat das InEK die Aufgabe, initial Krankenhäuser zu informieren, ob und welche pflegeintensive Abteilungen entsprechend den vorliegenden Fallzahlen aus 2017 vorhanden sind. Im Anschluss müssen die Krankenhäuser die betroffenen Stationen benennen. Hierfür stellt das InEK im Auftrag des Bundesministeriums für Gesundheit ein webbasiertes Erfassungsformular zur Verfügung. Hierfür wurde im InEK Datenportal der Bereich „Care“ geschaffen.

Dieses Dokument beschreibt die Maßnahmen zur Qualitätssicherung der genannten Funktionalität.

# Inhalt

[Qualitätsicherung Care (PpUGV) 1](#_Toc525815503)

[Überblick 1](#_Toc525815504)

[Inhalt 2](#_Toc525815505)

[1 Zielsetzung 3](#_Toc525815506)

[2 Programmierfehler 3](#_Toc525815507)

[2.1 Quellcode 3](#_Toc525815508)

[2.1.1 Statische Codeanalyse 3](#_Toc525815509)

[2.1.2 Code Review 3](#_Toc525815510)

[2.2 Unit-Test 3](#_Toc525815511)

[2.2.1 Unit-Test bei der Programmierung 3](#_Toc525815512)

[2.2.1 Unit-Test bei der Qualitätssicherung 4](#_Toc525815513)

[3 Mögliche Angriffsvektoren und deren Abwehr 4](#_Toc525815514)

[3.1 Unberechtigter Zugriff auf den HTTP-Server (DMZ) 4](#_Toc525815515)

[3.2 Direktzugriff auf die Datenbank 5](#_Toc525815516)

[3.3 Denial of Service (DoS) 5](#_Toc525815517)

[3.4 SQL-Injection 5](#_Toc525815518)

[3.5 HTTP und sonstige Injection 5](#_Toc525815519)

[3.6 Cross-Site Scripting und weitere Angriffe 6](#_Toc525815520)

[3.7 Unautorisierter bzw. direkter Aufruf einzelner Seiten 6](#_Toc525815521)

[3.8 Vulnerability Scan 6](#_Toc525815522)

[4 Look and Feel 6](#_Toc525815523)

[5 Termine 6](#_Toc525815524)

[Historie 8](#_Toc525815525)

# Zielsetzung

Das InEK stelle im Auftrag des Bundesministeriums für Gesundheit (BMG) ein Erfassungsformular für die Krankenhäuser mit pflegesensitiven Abteilungen zur Verfügung, über das die entsprechenden Daten eingesammelt werden. Aufgrund der politischen Bedeutung ist ein hoher Qualitätsstandard zu sichern.

Auch wenn die Fehlerfreiheit von Software nicht nachweisbar ist, wird das Risiko von Fehlern durch die entsprechenden und im Folgenden beschriebenen Maßnahmen minimiert. Dabei handelt es sich um Maßnahmen, wie sie bereits für das übrige Datenportal gelten; in weiten werden sie jedoch intensiver und mit zeitlich höherem Aufwand durchgeführt.

# Programmierfehler

Unter einem Programmierfehler wird eine Anweisung im Quelltext verstanden, welche zu einem unerwünschten Verhalten des Programms führt. Dies kann sich als offensichtliches Problem manifestieren, wie beispielsweise ein Programmabsturz, oder aber zu einer schleichend falschen Datenverarbeitung führen, die erst zu einem späteren Zeitpunkt auffällt.

## Quellcode

Beim Quellcode handelt es sich um Anweisungen in einer Programmiersprache, welche im Rahmen eines Build-Prozesses in eine ausführbare Applikation übersetzt werden.

### Statische Codeanalyse

Im Rahmen der Qualitätssicherung wird eine statische Code-Analyse durchgeführt. Dies dient der Suche nach eventuellen Fehlern und potenziellen Fehlerquellen, sowohl automatisiert als auch durch eine oder mehrere Personen, welche nicht den betreffenden Code entwickelt haben. Im Rahmen der Codeanalyse wird gleichzeitig die Qualität des Quellcodes im Hinblick auf Lesbarkeit, überschaubare Einheiten, Programmierstandards überprüft.

### Code Review

Im Rahmen eines Reviews stellt Entwickler den Quellcode einem anderen Entwickler oder einer Gruppe vor. Dabei wird gemeinsam auf die im vorherigen Abschnitt genannten Punkte geachtet.

## Unit-Test

Ein Unit-Test ist ein Stück Software, welche eine klar abgegrenzte kleine Programmeinheit, in der Regel einen Funktionsaufruf (in Java Methode genannt) überprüft. Unit-Tests werden beim Übersetzen des Quellcodes in das lauffähige Programm ausgeführt. Ändert sich das Verhalten des Programms versehentlich, so führt dies zu anderen Ergebnissen, als in den Unit-Tests definiert und es erfolgt eine entsprechende Fehlermeldung. Voraussetzung hierfür ist eine ausreichende Testabdeckung der kritischen Funktionen.

### Unit-Test bei der Programmierung

Unit-Tests werden direkt vom Entwickler zeitgleich oder sehr zeitnah mit den zu programmierenden Funktionen so erstellt, dass die Ausführung automatisch im Rahmen der Übersetzung in ein ablauffähiges Programm erfolgt.

### Unit-Test bei der Qualitätssicherung

Es wird geprüft, ob die Unit-Tests eine ausreichende Testabdeckung liefern. Sofern die Testabdeckung zu gering ist, erfolgt wahlweise ein Auftrag an den Entwickler zur Erstellung weiterer Tests oder die direkte Erstellung weiterer Tests im Rahmen der Qualitätssicherung-

# Mögliche Angriffsvektoren und deren Abwehr

Bereits im Rahmen früher Überlegungen wurde festgelegt, keine Daten in der DMZ zu speichern und auch keinen direkten Zugriff aus der DMZ auf die Datenbank zu erlauben. Dem System ist in der DMZ ein HTTP-Server vorgeschaltet, welcher gleichzeitig der Lastverteilung dient und Anfragen an das Datenportal auf einen von mehreren Applikationsservern weiterleitet. Der Zugriff auf die Daten erfolgt über einen dieser Applikationsserver, welche im InEK-Netz beheimatet sind.

Die Kommunikation mit dem Anwender erfolgt über einen verschlüsselten Kanal (HTTPS). Anfragen, die nicht für das Datenportal bestimmt sind, werden vom HTTP-Server nicht an die Applikation weitergeleitet. Anfragen im ungesicherten Protokoll werden auf das gesicherte Protokoll weitergeleitet. Nur mit diesem ist eine Nutzung des Datenportals möglich.

Dies vorausgeschickt, erfolgen hier weitere Überlegungen zu möglichen Angriffsvektoren sowie deren Abwehr.

Im Folgenden werden mögliche Angriffsvektoren gelistet. Daraus abgeleitet werden Maßnahmen, die entweder innerhalb oder außerhalb des Datenportals erfolgen. Soweit die Maßnahmen nicht innerhalb des Datenportals erfolgen, sind sie hier im Wesentlichen nur kurz erwähnt und werden durch die Administration umgesetzt. Maßnahmen innerhalb der Applikation haben dagegen direkten Einfluss auf die Programmentwicklung.

## Unberechtigter Zugriff auf den HTTP-Server (DMZ)

Sollte es einem Eindringling gelingen, Zugriff auf den HTTPS-Server oder gar Kontrolle über diesen zu erlangen, so besteht die Möglichkeit, diesen zu missbrauchen. Auf dem HTP-Server in der DMZ werden jedoch keine Daten vorgehalten, so dass ein möglicher Schaden entsprechend begrenzt ist. Vom HTTP-Server kann jedoch eine Verbindung mit dem Applikations-Server aufgenommen werden. Für diesen gelten weitere Sicherheitsmaßnahmen:

* Für den Zugriff auf ein Benutzerkonto ist die Kenntnis der Zugangsdaten (Login + Kennwort) erforderlich, so dass diese entsprechend geschützt sind.
* Zur Vermeidung eines Brute-Force-Angriffs werden die möglichen Loginversuche pro Zeiteinheit beschränkt.
* Der Zugriff auf die Daten ist auf geeignete Weise an ein erfolgreiches Login zu knüpfen
* Bei längerer Inaktivität erfolgt ein automatisches Logout

Vorrangiges Ziel ist jedoch, zu verhindern, dass eine unberechtigte Person überhaupt in das System eindringen kann. Dieses Ziel ist jedoch nicht über die Applikation, sondern über geeignete Maßnahmen, die für das gesamte Netz gelten, zu erreichen. Diese Maßnahmen sind von der Systemadministration festzulegen und durchzuführen. Sie liegen außerhalb des Fokus dieses Dokuments.

## Direktzugriff auf die Datenbank

Innerhalb des InEK-Netzes ist ein direkter Zugriff auf die Datenbank möglich. Dies ist erforderlich, damit die berechtigten Personen vernünftig mit diesen Daten arbeiten können, z.B. im Rahmen einer Überwachung des Regelbetriebs oder zur Beantwortung spezieller Anfragen seitens der Anwender. Da ein solcher Zugriff am Datenportal vorbei erfolgt, liegt dieser nicht im Fokus der Programmentwicklung und damit außerhalb dieses Dokuments

Es gelten die gleichen Schutzmaßnahmen wie für alle anderen Datenbanken:

Durch geeignete, von der Systemadministration festzulegende Maßnahmen, wird verhindert, dass Unberechtigte in das InEK-Netz eindringen oder sonst wie Zugriff auf die Daten (z.B. via Datensicherung) erhalten.

Für den Zugriff auf die Daten durch den Applikationsserver sowie berechtigte Personen werden nur die minimal erforderlichen Rechte vergeben.

## Denial of Service (DoS)

Bei diesem Angriff wird versucht, den Server durch eine hohe Anzahl von Zugriffen in einen betriebsunfähigen Zustand zu versetzen. Es handelt sich vornehmlich um einen Angriff auf den Server selbst und weniger auf die Applikation. Der vorgeschaltete HTTP-Server wird seitens der Administration entsprechend dem Stand der Technik geschützt.

Innerhalb der Applikation ist ein verstärkter Zugriff nicht im Sinne eines DoS, sondern zum Austesten von Logins denkbar. Dies ist weiter oben in "Unberechtigter Zugriff auf den HTTP-Server (DMZ)" beschrieben.

## SQL-Injection

Der Angreifer versucht, durch Eingabe von SQL-Ausdrücken in den Eingabefeldern die Kontrolle über das System zu erlangen oder Daten zu manipulieren.

Beispiel:

Das Login erfolge über ein dynamisch erstelltes SQL-Statement *… where password = ‘ + Eingabe + ‘*

Der Angreifer gibt im Kennwortfeld " ‘ or 1=1;“ ein.

Damit wird das Statement automatisch zu *… where password = ‘ ‘ or 1=1;‘* ergänzt und ein Login erfolgreich durchgeführt.

SQL-Injection wird im Rahmen der sicheren Programmentwicklung verhindert.

* Eingabefelder werden unmittelbar validiert, so dass sie nur gültigen Inhalt enthalten
* Parametrisierte Abfragen erfolgen ausschließlich via JPA und werden nicht wie im obigen Beispiel dynamisch zusammengebaut. Dies sorgt für eine sichere Erstellung der Statements entsprechend der sogenannten "Prepared Statements".
* An Stellen, an denen beispielsweise aus Performancegründen eine native Abfrage erforderlich ist, die dynamisch zusammengesetzt wird, erfolgt dies nur unter kontrollierten Bedingungen, z.B. Nutzung einer Ganzzahl-Variablen, welche eine textuelle Ergänzung wie oben angeführt, bereits durch ihren Datentyp verhindert.

Im Rahmen der Qualitätssicherung erfolgt ein besonderes Codereview im Hinblick auf die Datenanbindung-

## HTTP und sonstige Injection

Der Angreifer versucht, über die Eingabefelder HTTP-Code oder sonstiges, beispielsweise JavaScript-Code, in die Applikation einzuschleusen. Das Prinzip entspricht dabei der SQL-Injection; lediglich Sprachelemente und Ziel sind ein anderes. Die Abwehr dieser Bedrohung erfolgt in ähnlicher Art und Weise.

* Eingabefelder werden unmittelbar validiert, so dass sie nur gültigen Inhalt enthalten
* Eingabefelder werden „escaped“, so dass spezielle HTML-Zeichenfolgen nicht als solche interpretiert werden. Beispiel: „<b>“ als HTML interpretiert, führt zu einer Fettschrift. Soll dies als Text dargestellt werden, so wird dies als „&lt;b&gt;“ codiert. Die Sequenz „&..;“ codiert die Zeichen, so dass diese nicht als HTML interpretiert, für den Anwender aber korrekt dargestellt werden.

Das verwendete Web-Framework (JSF) sorgt automatisch für die korrekte Codierung, soweit dies nicht bewusst überschrieben wird. Es wird geprüft, ob nicht versehentlich das „Escapen“ abgeschaltet wurde.

## Cross-Site Scripting und weitere Angriffe

Bekannte Angriffsvektoren wie Cross-Site Scripting werden bereits durch das verwendete Web-Framework adressiert und entsprechend dem Stand der Technik abgewehrt, so dass hier keine besonderen Maßnahmen im Rahmen der Programmentwicklung erforderlich sind.

## Unautorisierter bzw. direkter Aufruf einzelner Seiten

Beim Datenportal und somit auch um die Formulare der PpUGV handelt es sich um eine Web-Applikation. Hierfür werden über einen Web-Browser einzelne „Seiten“ (URL) aufgerufen. Ohne ein vorheriges Login ist eine Nutzung der Applikation (mit Ausnahme öffentlicher Seiten wie Registrierung und Login) nicht zulässig.

Ruft ein Anwender versehentlich (oder Angreifer absichtlich) eine Seite ohne vorheriges Login auf, so erfolgt eine automatische Umleitung auf den Login-Dialog; eine nicht autorisierte Nutzung wird somit unterbunden.

Im Rahmen der Qualitätssicherung wird versucht, die Formulare der PpUGV ohne Anmeldung direkt aufzurufen. Erwartet wird bei jedem dieser Versuche die Umleitung auf den Login-Dialog.

## Vulnerability Scan

Im Rahmen der Qualitätssicherung werden toolgestützt automatisierte Scans durchgeführt. Die Tools sind in der Lage, die meisten der aufgeführten Angriffsvektoren durchzuführen und das Ergebnis zu dokumentieren.

# Look and Feel

Auch wenn die Funktionen entsprechend PpUGV technisch in die Infrastruktur des Datenportals eingebunden sind, handelt es sich um eine eigenständige Applikation, welche nach dem Login per einfachem Klick aufgerufen wird. Als solche ist sie optisch vom restlichen Portal derart abzugrenzen, dass ein Anwender in weiten Teilen das bekannte Look and Feel wiederfindet, die Applikation gleichzeitig direkt erkannt wird, beispielsweise durch eine andere Farbgestaltung.

Es muss unmittelbar erkennbar sein, dass das InEK die Erfassung im Auftrag des BMG durchführt.

# Termine

Die Qualitätssicherung erfolgt entwicklungsbegleitend. Somit gelten, maximal leicht zeitversetzt, die gleichen Terminvorgaben wie für die Entwicklung. Hierbei handelt es sich um

Bis 06.10.18 Alle Funktionen sind programmiert

Bis 11.10.18 Interner Anwendertest und Überarbeitung der Funktionen entsprechend Feedback

12.10.18 Die Funktionen PpUGV sind bereit zum Einsatz. Ggf. erfolgt zusätzlich ein externer Test

22.10.18 Versand der Informationsbriefe, Beginn Nutzung der PpUGV-Formulare

# Prüf-Ergebnisse

Die Prüfergebnisse dokumentieren die durchgeführten Prüfungen. Neben den o.a. Qualitätskriterien erfolgen auch funktionale und inhaltliche Prüfungen. Insofern können Reihenfolge und Inhalte von der obigen Aufzählung abweichen. Inhaltliche Prüfungen durch die Fachabteilung Ökonomie werden gesondert durchgeführt und sind nicht Bestandteil dieses Dokuments

## Look-And-Feel

Die Funktionalität Care (PpUGV) soll sich als eigenständiger Bereich optisch vom übrigen Datenportal absetzen, gleichzeitig gewohnte Strukturen beibehalten. Es ist als separate Applikation entwickelt, welche über das Daten-Portal-Login erreicht wird. Wie in den übrigen Portal-Bereichen findet der Anwender links eine Navigation, welche er zum Wechsel in andere Bereiche nutzen kann, während die eigentliche Dateneingabe im größeren Bereich rechts erfolgt. Diese Gemeinsamkeit sichert für existierende Portal-Nutzer eine weitgehend gewohnte Bedienung, wenn auch Care mit einer insgesamt moderneren Oberflächengestaltung daherkommt. Der Anwender findet sich hier schnell zurecht.

## SQL- und URL-Injection

### Datenbankzugriffe

Entsprechend statischer Codeanalyse der PpUGV-Datenzugriffe (Stand 09.10.18) gibt es drei Arten von Datenbankzugriffen, alle unter Verwaltung von JPA (Java Persistence API):

1. Direktes Mapping der Datenobjekte via JPA
2. Abfragen mittels JPQL (Java Persistence Query Language)
3. Abfragen mittels SQL (Structured Query Languages)

Das direkte Mapping (1.) ist in Bezug auf SQL-Injektion unkritisch.

JPQL-Abfragen (2.) werden durch JPA in SQL-Anweisungen übersetzt, so dass diese mit SQL-Abfragen (3.) gemeinsam betrachtet werden können.

Kritisch sind Funktionen, bei denen eine Zeichenkette (String) übergeben und in die SQL-Abfrage eingebaut wird, den der Anwender durch seine Eingaben vorgeben kann. In einen solchen String könnten maligne SQL-Anweisungen versteckt werden.

Derartige String-Übergaben sind nicht vorhanden. Damit ist eine SQL-Injection nicht möglich.

### Injection via URL

Zur Identifikation eines Datensatzes wird eine ID via URL angegeben. Eine URL ist grundsätzlich ein Text, so dass hier ein Angreifen versuchen könnte, die ID so zu manipulieren, dass hierüber eine SQL-Injection erfolgt. Letztendlich kann auch derart eingeschleuster Code nur beim Zugriff auf die Datenbank aktiv werden, so dass auch hier die oben festgestellten Schutzmechanismen greifen.

Im Rahmen der Qualitätssicherung wurde untersucht, ob über die URL-Injection neben SQL-Injection weitere Angriffe erfolgen könnten. Aus der URL wird der Paramater ID ausgelesen. Geprüft wurde, was passiert, wenn der Anwender hier manipulativ Werte vorgibt.

Ist der Anwender nicht angemeldet, so erfolgt – wie bei jeder nicht öffentlichen Seite des InEK Datenportals – eine Umleitung auf den Anmeldedialog. Somit ist es nur einem angemeldeten Anwender möglich, eine manipulierte URL zu nutzen.

Trägt der Anwender als ID die Kennung für neuen Datensatz ein, so kann er einen neuen Datensatz genauso anlegen, als ob er die „Neu“-Schaltfläche gewählt hätte. Das Anlegen eines Datensatzes erfolgt mit den Rechten des Anwenders. Insbesondere kann der Anwender nur ein freies IK gemäß seiner Berechtigung eintragen. Eine solch manipulierte URL kann der Anwender auch angeben, wenn die „Neu“-Schaltfläche nicht sichtbar ist, weil kein weiteres IK zur Verfügung steht. In einem solchen Fall kann der Anwender jedoch kein IK auswählen und damit den Datensatz auch nicht speichern. Damit ist eine derartig manipulierte URL unkritisch.

Handelt es sich bei der ID nicht um eine „Neu“-Kennzeichnung, so versucht das Programm diese in eine Ganzahl umzuwandeln. Manipuliert der Anwender die URL so, dass er hier eine SQL-Anweisung einträgt, so schlägt die Umwandlung in eine Ganzzahl fehl und der Anwender wird auf eine Fehlerseite geleitet. Somit ist weder SQL- noch eine andere Injection möglich.

Gibt der Anwender in der URL eine ungültige ID an, so wird er auf eine Fehlerseite geleitet.

Gibt der Anwender in der URL eine gültige ID an, so wird der Datensatz geladen, auch wenn der Anwender hierzu keine Berechtigung hat (Stand 10.10.18). An dieser Stelle ist noch das Standardverhalten zu implementieren: Nach Einlesen des ID prüft das Programm, ob der Anwender genügend Zugriffsrechte hat. Falls nicht, wird der Zugriff unterbunden. Damit wird eine solche Manipulation unkritisch.

Update 11.10.18: Das Standardverhalten ist implementiert. Gibt der Anwender manipulativ eine ID an, für die er keine Rechte hat, wird eine Fehlerseite angezeigt und der Anwender ausgeloggt.

Eine Web-Applikation wird über URL gesteuert. Insofern lassen sich Manipulationen nicht ausschließen. Durch die geprüften Maßnahmen – wenn das Standard-Verhalten implementiert wurde – sind fehlerhafte Zugriffe durch solche Manipulationen gemäß Stand der Technik auszuschließen.

Auch der Versuch, SQL-Injection automatisiert via Penetrationstool einzuschleusen, ist erwartungsgemäß fehlgeschlagen.

## Logging von Nutzeraktivitäten

Es erfolgt ein Aktivitäten log, das heißt Aktivitäten wie Speichern, Senden etc. werden dokumentiert. Hierzu werden alle Speicherzeitpunkte mit dem jeweiligen Bearbeitungsstatus in ein Log geschrieben. Wechselt beispielsweise der Status von „neu“ auf „gesendet“, so kennzeichnet dies den Sendezeitpunkt an das InEK.

Der Eintrag in das Action-Log wurde positiv überprüft.

Ein ChangeLog, in dem jede Feldänderung mit Feldwerten aufgezeichnet wird, ist bei Care nicht implementiert.

## Statische Codeanalyse

Im Rahmen der Codeanalyse wurden diverse kleinere Schwachpunkte entdeckt und behoben.

Das Datenmodel reicht in der Hauptsache Daten ohne weitere Verarbeitung von der Datenbank zur Oberfläche durch. Eine weitergehende Verarbeitung – und damit ein Ansatz für Unit-Tests – ist nur sporadisch vorhanden. Derzeit (11.10.18) existiert nur ein Unit-Test.

Historie

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Datum** | **Bearbeiter** | **Änderung** |
| 0.1 | 2018-09-27 | MM | Initiale Erstellung |
| 1.0 | 2018-10-09 | MM | Prüf-Ergebnisse |
| 1.1 | 2018-10-10 | MM | Prüf-Ergebnisse |
| 1.2 | 2018-10-11 | MM | Prüf-Ergebnisse |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |