Analysebericht Nr. NA07/23 zur Server-Software „KSV-Backend“

von David Bouck-Standen, M.Sc.

Auftraggeber:
KSV-Fuchs
Vorläufiger Entwurf

Inhalt

Einleitung

Diese Einleitung befasst sich mit der Vorstellung des Autors dieses Berichts und seiner Qualifikation, einer kurzen Beschreibung des Prüfobjekts und des Auftrags sowie der Zielsetzung der hier durchgeführten Prüfung.

Autorenvorstellung

Der vorliegende Analysebericht zur Software, wurde sorgfältig von David Bouck-Standen erstellt. Mit über zwei Jahrzehnten tiefgreifender Erfahrung in der Backend-Software-Entwicklung und einer Position als Senior Software-Systems Engineer in einem multinationalen IT-Unternehmen, verbindet David Bouck-Standen praktische Expertise mit akademischer Brillanz.

Mit einem Abschluss eines Bachelor of Science in Informatik einer renommierten deutschen Universität spezialisierte sich der Autor auf Medieninformatik mit einem besonderen Fokus auf Mensch-Maschine-Systeme, außerdem beendete der Autor sein Bachelor-Studium mit Medizininformatik mit einem zweiten, zusätzlichen Anwendungsbereich. Im weiterführenden Studium mit Abschluss Master of Science in Informatik einer deutschen Universität wählte der Autor den Schwerpunkt Verteilte Informationssysteme. Dieses Spezialgebiet zeichnet sich durch umfassende Anforderungen an Systemarchitekturen, Datenschutz und Datensicherheit aus, was es zu einem außerordentlich spezialisierten Bereich macht. Auch über die sich an das Studium anschließende mehrjährige wissenschaftliche Tätigkeit des Autos hinaus publiziert dieser fortwährend auf internationalen Konferenzen. Seine Arbeiten werden international anerkannt und mehrfach international ausgezeichnet, insbesondere im Bereich semantischer kontextualisierter Modellierungen über verteilten Informationssystemen.

Heute steht David Bouck-Standen neben seiner Tätigkeit als Senior Software-Systems-Engineer zahlreichen Unternehmen als externer Datenschutzbeauftragter zur Seite und bietet Beratung im Bereich Datensicherheit. Er hat eine besondere Expertise in der Analyse von Soft- und Hardware hinsichtlich Datenschutz und Datensicherheit entwickelt und ist in diesem Bereich eine anerkannte Autorität.

Beschreibung des Prüfobjekts

Bei dem vorliegenden Prüfobjekt handelt es sich um eine in der Programmiersprache C# implementierte Software-Lösung, die im Backend-Bereich als Server-Lösung betrieben werden soll. Das vorliegende Programm erhält über eine separate, nicht zum Prüfobjekt gehörende Softwareschnittstelle (auch User-Interface genannt) die Eingaben von Benutzern zur weiteren Verarbeitung.

Inhaltlicher Gegenstand des Prüfobjekts ist die Schaffung einer Schnittstelle zwischen einer grafischen Benutzungsschnittstelle, über die Benutzer (hier: *Betroffene* gem. DSGVO) Daten der Kategorien (i) personenbezogene Daten zur *direkten Identifikation* der Benutzer sowie (ii) wirtschaftliche Daten in Form von Finanz- bzw. Kreditdaten erfassen, und in diesen Kategorien weiterführende Informationen oder Daten übermitteln. Diese Daten sind als besonders schutzwürdig einzustufen. Möglicherweise enthalten eingereichte Informationen noch Daten weiterer Kategorien, wie etwa Identifikationsnummern. Die Daten dieser Benutzer werden vom Prüfobjekt digital gespeichert und weiterverarbeitet. Das Ergebnis der Weiterverarbeitung wird

wiederum in der Benutzungsschnittstelle visualisiert und bereitgestellt. Weitere Benutzer des Prüfobjekts (hier: nicht Betroffene i.S.d. DSGVO) sind Mitarbeiter und Administratoren, die das Prüfobjekt als Werkzeug zur (teil-)automatisierten Verarbeitung der übermittelten Daten gegenständlich zur Erbringung ihrer Dienstleistung als Kreditsachverständige einsetzen.

Das Prüfobjekt wurde vorgelegt in Form einer Software-Lösung die mit der Programmierumgebung Microsoft Visual Studio entwickelt wurde, und in der das vorliegende Prüfobjekt in Form von Quellcode gespeichert ist. Das Projekt kompiliert, d.h. die Software ist grundsätzlich in einem ausführungsbereiten Zustand, vorbehaltlich korrekter Konfiguration eines entsprechenden Server-Systems, z.B. mit entsprechend bereitgestellter Datenbank und Anbindung eines diese Backend-Lösung bedienenden Frontends (Benutzungsschnittstelle).

Zielsetzung der Prüfung

Das Ziel der Prüfung des Prüfobjekt ist es, den Quellcode hinsichtlich des Ein- und Ausgangs von (personenbezogenen) Daten zu analysieren, d.h. im Einzelnen zu klären, warum, wo, wohin und wann personenbezogene Daten innerhalb des Prüfobjekts verarbeitet, gespeichert und oder übertragen werden. Dabei beschränkt sich dieser Bericht auf Sachverhalte, die mit der Verwendung der Daten in Zusammenhang stehen. Folgende Teilfragen sind diesbezüglich zu klären:

- Kann die vorliegende Software uneingeschränkt gem. DSGVO betrieben werden?
- Werden (personenbezogene) Daten durch die Software während des Betriebs auf andere Systeme, als die durch den Auftraggeber betriebenen, übertragen?

Auftragsbeschreibung

Mit Hilfe der mit dem Prüfobjekt vorliegenden Backend-Software strebt der Auftraggeber die (teil)automatisierte Verarbeitung sensibler Informationen seiner Kunden (Benutzer, genderneutral) an. Es ist beabsichtigt, dass die Benutzer über die Software personenbezogene Daten an den Auftraggeber übermitteln, die u.a. Informationen zu Krediten und Kreditgeschäften der Benutzer enthalten, die der Auftraggeber in der Ausübung seines Berufs als Kreditsachverständiger i.S.d. DSGVO *verarbeitet*.

Aufgrund des im Vorabsatz genannten Einsatzszenarios der Software ist der Auftraggeber bestrebt, dass sämtliche Datenverarbeitung nach aktuellem Stand der Technik sicher durchgeführt werden kann. Als *sicher* ist hier, neben der intrinsischen Systemsicherheit hinsichtlich der Ausführung der Software – etwa der Vermeidung des Datenverlusts während der Ausführung von Berechnungen, der Speicherung, oder der sonstigen Verarbeitung – oder der Datenintegrität – etwa durch Implementierung geeigneter Sicherungsstrategien – vor allem aber die Datensicherheit hinsichtlich des Schutzes der Daten vor möglichen Zugriffen durch Dritte definiert.

Neben den voranstehend bezeichneten Funktionalitäten werden mit dem Prüfobjekt außerdem Aufträge des durch den Auftraggeber betriebenen Shops (lt. Auftraggeber z.B. für den Verkauf eines Buches) verarbeitet.

Der Autor dieses Berichts wurde damit beauftragt, den vorliegenden Programmcode (auch als *Quellcode* bezeichnet) der Software Zeile für Zeile darauf zu prüfen, ob personenbezogene Daten, die Benutzer in das System übermitteln, an andere Systeme (Dritte) übermittelt werden, als die Systeme, die der Auftraggeber selbst betreibt. Zur Umsetzung des Prüfvorhabens sind zusätzlich sämtliche Programmbibliotheken zu prüfen, die durch die vorliegende Software genutzt werden. Programmbibliotheken bündeln Funktionalitäten, die nicht erneut implementiert werden müssen, und aus dieser Bibliothek genutzt werden können. Häufig stellen auch Anbieter bestimmte Dienstleistungen in Form von Bibliotheken bereit, sodass diese Dienstleistungen unter Nutzung der Schnittstellen der Bibliotheken in andere Programme quasi „nahtlos“ integrierbar sind. Für gewöhnlich ist es nicht möglich, den genauen Programmcode von externen Bibliotheken einzusehen, da diese durch die jeweiligen Anbieter implementiert und ggf. vor dem Zugriff durch Dritte geschützt werden. Dies kann unter Umständen ein Risiko bei der Verwendung solcher Bibliotheken darstellen, gerade wenn diese unerwünschte Funktionalitäten ausführen, die neben den möglicherweise beworbenen, beabsichtigt genutzten Funktionalitäten im Verborgenen ausgeführt werden.

Diese Analyse beschränkt sich hinsichtlich des vorliegenden Quellcodes auf die Untersuchung eines Daten-Ausgangs an Dritte. Das Programm wird nicht zur Ausführungs- oder Laufzeit untersucht. Andere Programme[1], die ihrerseits Zugriff auf die durch das Prüfobjekt erstellten Daten nehmen, sind nicht Gegenstand dieser Analyse.

[1] Hier: z.B. das durch den Auftraggeber entwickelte Programm ALF.

Bezüglich einer Prüfung von genutzten externen Bibliotheken beschränkt sich diese Analyse auf die von den jeweiligen Entwicklern bereitgestellte Dokumentation und die Risikoeinschätzung des Autors hinsichtlich der jeweiligen Verwendung.

Diese Analyse führt keine Prüfung hinsichtlich der Lizenzen oder Lizenzierung eingesetzter interner oder externer Bibliotheken vor und prüft oder bewertet nicht, ob sich diese im Allgemeinen oder im Speziellen für den beabsichtigten Einsatz eignen.

Beschreibung des Prüfobjekts

Das Prüfobjekt liegt in Form einer Visual Studio Lösung (Entwicklungsumgebung) und den dort enthaltenen Programmteilen, sogenannten „Klassen“ vor, die in der Programmiersprache C# implementiert wurden und nach dem Schema einer objektorientierten Programmierung erstellt wurden. Die Lösung enthält außerdem eine Bibliothek-Beschreibungsdatei, mit deren Hilfe eine Liste externer Bibliotheken durch die Entwicklungsumgebung aus öffentlich zugänglichen Quellen geladen wird. Der vollständige Quellcode liegt in der Anlage dieses Berichts vor.

Projektsichtung

Das Prüfobjekt strukturiert sich in ein (serverseitig) ausführbares Programm, das komponentenweise entwickelt wurde. Es ist festzustellen:

- Die Implementierung ist klar konzeptioniert und strukturiert.
- Die Implementierung verfügt über eine rein dürftige Quellcode-Kommentierung und folgt damit nicht den Empfehlungen des Framework-Herstellers Microsoft[2], die in der Industrie allseits anerkannt werden.
- Zum vorliegenden Programm ist keine Dokumentation verfügbar oder als Anlage an das Software-Projekt gereicht.
- Einer durch die Entwicklungsumgebung automatisiert erstellten Dokumentation mangelt es an wesentlichen Informationen, da implementierte Klassen und Methoden, wie auch der übrige Quellcode, größtenteils und abweichend der Empfehlungen des Herstellers (s.o.) nicht mit entsprechenden Klassen- und Methodenkommentaren versehen wurden, die Auskunft über eine Parametrisierung, Invariante, Sinn, Zweck oder Funktionsweise geben. Daraus folgt, dass auch eine automatisiert erstellte Quellcode-Dokumentation größtenteils ohne deskriptiven Inhalt resultieren würde. Auch diese quellcodenahe Dokumentation müsste noch erstellt bzw. mit Hilfe der Entwicklungsumgebung generiert werden, und wird nicht als Anlage an das Projekt gereicht.

Externe Bibliotheken

Das Prüfobjekt erfordert die Nutzung der folgenden externen Bibliotheken, ohne die dieses nicht ausgeführt werden kann, wobei sich diese Liste auf Bibliotheken beschränkt, die nicht ohnehin als Bestandteile der zugrundeliegenden Entwicklungsumgebung gezählt werden:

- **ClosedXML** ist eine Open-Source .NET-Bibliothek, die das Lesen, Manipulieren und Schreiben von Excel-Dateien im Format 2007+ (.xlsx, .xlsm) ermöglicht. Sie wurde entwickelt, um eine höhere Abstraktionsebene über dem OpenXML SDK zu bieten, um das Arbeiten mit Excel-Dateien zu vereinfachen. Hier sind einige ausgewählte Merkmale von ClosedXML, die zum besseren Verständnis dienen sollen:
    - **Einfachheit:** Im Gegensatz zur direkten Arbeit mit dem OpenXML SDK, das eine detaillierte Kenntnis der OpenXML-Spezifikation erfordert, bietet ClosedXML eine intuitive API, mit der Benutzer schnell und einfach Excel-Dateien bearbeiten können.
    - **Reichhaltige Funktionen:** Mit ClosedXML können Benutzer eine Vielzahl von Aufgaben durchführen, einschließlich dem Hinzufügen und Löschen von

[2] Microsoft: *Common C# Code Conventions.* 2023. Abgerufen: 02.09.2023 (https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions).

Arbeitsblättern, dem Manipulieren von Zellen und Zeilen, dem Setzen von Stilen, dem Erstellen von Formeln und vielen anderen.
- **Kein Excel erforderlich:** Da ClosedXML direkt mit der OpenXML-Spezifikation arbeitet, ist es nicht erforderlich, dass Microsoft Excel auf dem Computer installiert ist, um die Bibliothek zu verwenden.
- **Plattformübergreifend:** Da es sich um eine .NET-Bibliothek handelt, kann ClosedXML sowohl auf dem .NET Framework als auch auf .NET Core verwendet werden, was bedeutet, dass es auf verschiedenen Betriebssystemen wie Windows, macOS und Linux funktioniert.

- **coverlet.collector**: ist ein Teil des Coverlet-Projekts, das für .NET eine Code-Coverage-Lösung bietet. Code-Coverage-Tools messen, welcher Teil des Codes während der Ausführung von Tests tatsächlich ausgeführt wird, was den Entwicklern hilft zu verstehen, welche Teile des Codes nicht getestet werden. Hier sind einige ausgewählte Schlüsselpunkte zu coverlet.collector, die zum besseren Verständnis dienen sollen:
  - **Plattformübergreifend**: Coverlet unterstützt sowohl .NET Framework als auch .NET Core, was bedeutet, dass Entwickler Code-Coverage für Projekte auf verschiedenen Plattformen erhalten können.
  - **Integration in Test-Frameworks**: Coverlet kann in beliebte Test-Frameworks wie xUnit, NUnit und MSTest integriert werden.
  - **Verschiedene Ausgabeformate**: Coverlet kann Ergebnisse in verschiedenen Formaten wie Cobertura, Icov, OpenCover und anderen ausgeben.
  - **Visual Studio-Integration**: Mit coverlet.collector können Entwickler Code-Coverage-Daten direkt im Test-Explorer der Entwicklungsumgebung Visual Studio sammeln, ohne ein separates Tool verwenden zu müssen.

- **Sentry.AspNetCore** ist die offizielle Integration von Sentry für ASP.NET Core-Anwendungen. Sentry selbst ist eine Open-Source- und gehostete Fehlerverfolgungsplattform, die Entwicklern hilft, Abstürze, Fehler und Performance-Probleme in ihren Anwendungen in Echtzeit zu überwachen und zu beheben. Diese Bibliothek bietet spezifische Funktionen und Integrationen für ASP.NET Core-Anwendungen. Mit dieser Integration können Entwickler:
  - **Echtzeit-Fehlerberichte erhalten**: Wenn in einer ASP.NET Core-Anwendung ein Fehler auftritt, wird dieser automatisch an Sentry gesendet, wo er analysiert, aggregiert und dem Entwickler zur Überprüfung präsentiert wird.
  - **Detaillierte Fehlerkontexte sehen**: Sentry kann detaillierte Informationen über Fehler bereitstellen, einschließlich Stack-Traces, Anforderungsinformationen, beteiligte Benutzer und mehr.
  - **Performance-Monitoring**: Abgesehen von Fehlerberichten bietet Sentry auch Tools zum Überwachen der Anwendungsleistung, was Entwicklern hilft, Flaschenhälse und Performance-Probleme zu identifizieren.
  - **Benutzerdefinierte Daten senden**: Entwickler können zusätzliche Daten an Sentry senden, um den Kontext von Fehlern oder Ereignissen zu erweitern, was bei der Fehlerdiagnose hilfreich sein kann.

- **Integration mit anderen Systemen**: Sentry bietet Integrationen mit vielen anderen Tools und Diensten, sodass Entwickler Benachrichtigungen über verschiedene Kanäle erhalten oder die Daten in andere Systeme exportieren können.

- **sib_api_v3_sdk** ist die offizielle SDK (Software Development Kit) für SendinBlue's RESTful API Version 3, spezifisch für die C#-Sprache. SendinBlue ist eine Online-Marketingplattform, die Tools für E-Mail-Marketing, SMS-Kampagnen und andere ähnliche Dienstleistungen anbietet. Mit dieser SDK können Entwickler:
  - **Automatisierung**: Automatisieren Sie das Senden von E-Mails, das Verwalten von Kontakten, das Erstellen von Kampagnen und andere Funktionen, die SendinBlue bietet, direkt aus ihren C#-Anwendungen heraus.
  - **Integration**: Integrieren Sie SendinBlue-Dienste in bestehende Anwendungen, Systeme oder Webseiten.
  - **Zugriff auf alle API-Funktionen**: Die SDK bietet Zugriff auf nahezu alle Funktionen, die über die SendinBlue API verfügbar sind, einschließlich, aber nicht beschränkt auf das Senden von Transaktions-E-Mails, das Verwalten von Kontakten, das Abrufen von Berichten und mehr.

  Für Entwickler, die SendinBlue in ihre C#-Anwendungen oder -Dienste integrieren möchten, erleichtert die SDK den Prozess erheblich, da sie bereits Methoden und Funktionen bereitstellt, um mit der API zu kommunizieren, ohne dass man den gesamten Code von Grund auf neu schreiben muss.

- **Swashbuckle.AspNetCore** ist eine Bibliothek für .NET Core-Anwendungen, die Werkzeuge zur Erstellung von Swagger- und OpenAPI-Dokumentationen für ASP.NET Core-Web-APIs bietet. Swagger (heute oft als OpenAPI bezeichnet) ist ein Spezifikationsstandard zur Beschreibung, Produktion, Konsumierung und Visualisierung von RESTful Webdiensten. Mit dieser Bibliothek können Entwickler:
  - **Automatische API-Dokumentation generieren**: Die Bibliothek kann die Struktur und die Endpunkte einer Web-API analysieren und automatisch eine Swagger- oder OpenAPI-Spezifikationsdatei generieren.
  - **Swagger UI bereitstellen**: Swashbuckle enthält auch eine Integration für Swagger UI, eine interaktive Web-Oberfläche, mit der Benutzer die API-Dokumentation visualisieren und interaktiv mit der API interagieren können.
  - **Client-Code-Generierung unterstützen**: Mit einer gültigen OpenAPI-Spezifikation können Entwickler Client-Code in verschiedenen Programmiersprachen generieren, um die API leichter konsumieren zu können.
  - **API-Versionierung**: Die Bibliothek bietet auch Unterstützung für API-Versionierung, so dass Sie verschiedene Versionen Ihrer API effektiv dokumentieren können.

  Diese Bibliothek hat sich als beliebtes Tool in der .NET Core-Community etabliert, da es den Prozess der Dokumentation und Interaktion mit Web-APIs erheblich vereinfacht. Es ist besonders nützlich in größeren Teams oder in öffentlichen APIs, wo eine klare und konsistente Dokumentation für die effektive Nutzung der API unerlässlich ist.

- **xUnit.net** ist ein Test-Framework für .NET, das von Entwicklern verwendet wird, um sog. Unit-Tests für ihre Softwareprojekte zu schreiben und auszuführen. Es verfügt über zahlreiche für Softwareentwickler relevante Funktionen zur Sicherung der Qualität von implementierter Software.

- **xunit.runner.visualstudio** ist ein Test-Runner-Plugin für xUnit.net, das speziell für die Integration mit der Entwicklungsumgebung Microsoft Visual Studio entwickelt wurde. Es ermöglicht Entwicklern, xUnit.net-Tests direkt in der Visual Studio IDE auszuführen und die Ergebnisse zu sehen, ohne eine separate Tool- oder Befehlszeilenumgebung verwenden zu müssen.

Weitere externe Schnittstellen

Neben den direkt als externe Module erkennbaren Bibliotheken werden weitere externe Schnittstellen genutzt, die im Folgenden betrachtet werden.

1. **WooCommerce**: Im Prüfobjekt kommt außerdem die Nutzung der API des Online-Shop-Systems WooCommerce zum Einsatz. Rein hypothetisch darf hier angenommen werden, dass dieses in Zusammenhang mit der vom Auftraggeber betriebenen Webseite im System WordPress steht, die frei im Internet verfügbar ist[3].

   Mit WooCommerce können Anbieter, wie etwa hier der Auftraggeber, physische und digitale Produkte verkaufen, verschiedene Versand- und Zahlungsoptionen konfigurieren, Steuersätze verwalten und viele weitere Funktionen nutzen, die für den Betrieb eines Online-Geschäfts erforderlich sind.

   WooCommerce wurde erstmals 2011 veröffentlicht und hat sich seitdem zu einer der dominierenden E-Commerce-Plattformen im Internet entwickelt, insbesondere für kleinere bis mittelgroße Unternehmen und Einzelpersonen, die einen flexiblen und kostengünstigen Weg suchen, um Produkte online zu verkaufen. Es wird nun von Automattic betreut, dem Unternehmen hinter WordPress.com.

   Neben WordPress[4] kann die WooCommerce-API auch durch andere Programme angesprochen werden, die dann die dortigen Funktionalitäten nutzen können.

[3] Anmerkung des Autors.
[4] WordPress ist ein Online-CMS-System, vgl. wordpress.com.

Methodik

Zur Erstellung dieses Berichts werden zunächst die externen Bibliotheken des Programms untersucht und hinsichtlich einer möglichen Weitergabe von Daten an Dritte hin evaluiert. Im Anschluss wird der vorliegende Programmcode des Prüfobjekts Zeile für Zeile durchgesehen und die Funktionsweise nachverfolgt, um die Eingangs definierten Fragen unter den diesem Bericht zugrunde liegenden Einschränkungen zu beantworten.

Zur Durchsicht des Programmcodes wird das Programm durch den Autor in die Entwicklungsumgebung Microsoft Visual Studio 2022 geladen. Dies ermöglicht neben der zeilenweisen Prüfung auch, Querverweise, Objektbeziehungen, Datenherkunft, Variablenbelegungen, usw. und außerdem programmiertechnische Zusammenhänge nachzuvollziehen.

Externe Bibliotheken werden zunächst anhand der Dokumentation des jeweiligen Herstellers auf ihre Funktionalität hin geprüft und anschließend im Zusammenhang mit dem Einsatz im Prüfobjekt evaluiert. Externe Bibliotheken, die nach aktuellem Stand der Technik oder dem Erfahrungsstand des Autors im Allgemeinen als unbedenklich einzustufen sind, werden nicht näher untersucht. Hierzu zählen beispielsweise die grundlegenden Bibliotheken, aus denen das für das Prüfobjekt verwendete Programmierframework besteht und auf dessen Basis das Prüfobjekt entwickelt wurde. Steht für externe Bibliotheken kein offener Quellcode zur Überprüfung zur Verfügung, werden diese als kritisch eingestuft.

Befund

Zunächst werden die im Prüfobjekt referenzierten Bibliotheken einer Prüfung unterzogen, des Weiteren der vorliegende Quellcode. Bibliotheken, die hier nicht verzeichnet sind, werden vom Autor als potentiell sicher eingestuft. Bibliotheken mit dem Präfix „BCF“ werden in der erweiterten Quellcodeprüfung im Abschnitt „nuget.config“ betrachtet.

Prüfung der Bibliotheken

Im Folgenden werden die Bibliotheken sowie ihre Verwendung im Hinblick auf die Prüffragen einer Prüfung unterzogen.

ClosedXML

Die Bibliothek ClosedXML wird im Zusammenhang mit der durch das Prüfobjekt durchgeführten Datenaufbereitung genutzt und dazu verwendet, um Excel-Dateien zu erstellen bzw. auszulesen (z.B. *CreateExcelForAlfService.cs*, Zeile 766). Die Daten werden u.a. auf die Festplatte des Servers geschrieben, um diese für das externe Programm ALF nutzbar zu machen, welches der Auftraggeber anderweitig entwickeln ließ und nach eigener Aussage auf dem gleichen Server betreibt, wie das Prüfobjekt.

Erfahrungsgemäß stellt die ordnungsgemäße Verwendung dieser Bibliothek kein Risiko für die Anwendung im Zusammenhang mit der Prüffrage dar. Auch aus der vorliegenden Implementierung folgt kein ableitbares Risiko in der beabsichtigten oder tatsächlichen Verwendung.

Ein unmittelbares Risiko im Zusammenhang mit Datensicherheit und Datenintegrität kann bestehen, wenn für Verzeichnisse, die durch ClosedXML (oder ALF) genutzt werden, Lese- und oder Schreibrechte bestehen, die einen unberechtigten Zugriff durch Dritte zulassen. Dies ist beispielsweise, möglicherweise nicht abschließend aufgezählt, unter folgenden Umständen möglich:

- Ein Verzeichniszugriff entsteht indirekt durch fehlerhafte Serverkonfiguration, bei der Leserechte auf das Verzeichnis gesetzt werden, die unabhängig der Anwendungsebene bestehen, und Verzeichnisinhalte über HTTP/HTTPS lesbar machen.
- Ein Verzeichniszugriff über eine SSH-Schnittstelle ermöglicht wird, bei dem entweder anonym oder für einen unberechtigten Benutzer ein Lese- oder Schreibzugriff auf das entsprechende Verzeichnis ermöglicht wird.

Sentry.AspNetCore

Die Bibliothek Sentry.AspNetCore wurde speziell entwickelt, um Fehler- und Leistungsdaten an den Sentry-Dienst zu senden. Dies ist der primäre Zweck von Sentry: Es sammelt, analysiert und präsentiert Fehler, Abstürze und Leistungsinformationen von Ihren Anwendungen, damit Sie diese Probleme beheben können.

Wenn ein Fehler in Ihrer ASP.NET Core-Anwendung auftritt oder ein Leistungsereignis erfasst wird, sendet Sentry.AspNetCore Details zu diesem Fehler oder Ereignis an die Sentry-Server, die entweder gehostet (z.B. auf sentry.io) oder in Ihrer eigenen Infrastruktur betrieben werden können, wenn Sie eine selbst gehostete Version von Sentry verwenden.

Sentry wird im vorliegenden Programmcode offenbar zum Loggen von Nachrichten verwendet zu. Geloggt wird hierbei aber zu einer externen Internetadresse, die mangels hier zum Prüfungszeitpunkt vorliegender Auftragsverarbeitungsvereinbarung des offensichtlichen Betreibers der unter der Domäne „sentry.braodcastx.io“ als außerhalb des Einflussbereichs des Auftraggebers zu bewerten ist. In den Applikationseinstellungen, die jeweils bei Start des Programms geladen werden und in der Datei *appsettings.json* verzeichnet sind, ist hier der Datenempfänger spezifiziert unter „https://7209d1fa5cb24fa2a58a2f528b5571ac@sentry.broadcastx.io/34“ (Zeilen-Trennungszeichen zur besseren Lesbarkeit durch den Autor eingefügt) (vgl. *appsettings.json*, Zeile 49).

An bzw. mit Sentry werden im vorliegenden Prüfobjekt über reine Debug-Informationen, die während einer Entwicklung des Systems durch Entwickler genutzt werden, hinausgehend Informationen an den Betreiber des im Vorabsatz beschriebenen Systems übermittelt. Dabei handelt es sich um die komplette „ShopOrder“, als den vollständigen Datensatz eines Bestellvorgangs im über *WooCommerce* betriebenen Shop des Auftraggebers, die serialisiert als sog. Json textuell personenbezogene Daten enthält. (vgl. *KsvShopOrdersController.cs*, Zeile 120).

SendInBlue/sib_api_v3_sdk

Die *sib_api_v3_sdk* ist die offizielle SDK für SendinBlue's RESTful API Version 3 für C#. Ihr primärer Zweck ist es, Entwicklern die Interaktion mit den Diensten von SendinBlue zu erleichtern. Das bedeutet, dass sie Daten an die SendinBlue-Server überträgt, da das ihr Hauptzweck ist.

Wenn SendinBlue-Dienste über diese SDK genutzt werden, werden personenbezogene Daten (wie E-Mail-Adressen, Nachrichteninhalte, Kampagnendaten usw.) an SendinBlue gesendet. Obwohl SendInBlue nicht der beabsichtigte Empfänger dieser Daten ist, kann dieser Dienstleister rein hypothetisch auf z.B. Nachrichteninhalte zugreifen.

Im Prüfobjekt werden Daten über SendinBlue-Dienste versendet:

1. Aktivierungs-, Registrierungs- und Zugangsdaten von Benutzern und solche E-Mail-Nachrichten, die sich auf diese Daten beziehen.
2. Personenbezogene Daten von Benutzern (vgl. *SendInBlueMailService.cs,* Zeile 127).

Grundsätzlich ist gegen den Versand von Nachrichten zu (1) nichts einzuwenden, da dies auch die allgemeine Praxis für die Bereitstellung von Zugängen zu Online-Systemen darstellt. Bei Bank-, Finanz- und Wirtschaftsdaten sollte zusätzlich eine 2-Faktor-Authentifizierung mit einem zusätzlichen Kanal genutzt werden[5], etwa per SMS-Token, 2-Faktor-App oder vergleichbar. In diesem Zusammenhang stellt die Nutzung eines externen Dienstleisters für den Nachrichtenversand durch das Prüfobjekt ein Sicherheitsrisiko dar, da E-Mail-Inhalte für Dritte ersichtlich werden. Dies geschieht auch völlig unnötigerweise, da der E-Mail-Versand und Versand von Benachrichtigungen auch direkt aus dem Prüfobjekt erfolgen könnte.

[5] D.h.: PIN per E-Mail zu versenden wäre keine valide Option, da der Kommunikationskanal äquivalent der übrigen Kommunikation von Benutzer mit System ist, und dadurch keine zusätzliche Hürde für Angreifer entstehen würde.

Der Versand von Nachrichten zu (2) stellt ein erhebliches Risiko dar. Der E-Mail-Versand erfolgt unverschlüsselt, insofern ist es Dritten möglich, ganz ohne Systemzugriff auf das Prüfobjekt (oder die Systeme des Benutzers) auf dem Kommunikationswege Zugriff auf die Daten des Benutzers zu nehmen, die das Prüfobjekt versendet. Darüber hinaus werden die Daten von SendinBlue verarbeitet, sodass auch diese Partei potentiell Zugriff auf die Daten von Benutzern nehmen könnte. Hier empfiehlt es sich, diese Daten lediglich nach Login in das Prüfobjekt und dort zur Anzeige oder zum Download anzubieten, da nur so gewährleistet werden kann, dass der Übertragungsweg zwischen Prüfobjekt und Web-Browser des Benutzers ausreichend gesichert ist.

Swashbuckle.AspNetCore

Die Bibliothek Swashbuckle.AspNetCore bietet in erster Linie ein Werkzeug zur Erstellung und Bereitstellung von API-Dokumentationen für ASP.NET Core-Webdienste mittels der Swagger/OpenAPI-Spezifikation. Es selbst sammelt oder sendet keine Daten an externe Dienste oder Dritte.

Wird diese Bibliothek in einer Anwendung verwendet, wird eine Swagger UI-Oberfläche und eine zugehörige OpenAPI-Spezifikationsdatei generiert. Diese Oberfläche und Spezifikation basiert auf dem Code der jeweiligen Anwendung und den in Ihrem Code definierten API-Endpunkten. Hier sind jedoch zu beachten:

1. **Zugriffskontrolle**: Je nach Konfiguration und Einsatzumgebung kann die von Swashbuckle generierte Dokumentation öffentlich zugänglich sein. Es ist wichtig, sicherzustellen, dass keine sensiblen Informationen oder interne Details über die Anwendung oder das System preisgegeben werden.
2. **Dokumentationsinhalte**: Es ist darauf zu achten, welche Informationen in der API-Dokumentation angezeigt werden. Details wie Datenbankstrukturen, interne Systeminformationen oder andere vertrauliche Informationen dürfen nicht öffentlich geteilt werden.
3. **Zusätzliche Erweiterungen oder Plugins**: Wenn zusätzliche Erweiterungen oder Plugins in Kombination mit Swashbuckle verwendet werden, sind außerdem die jeweiligen Dokumentationen und Datenschutzrichtlinien zu prüfen, um sicherzustellen, dass keine unerwünschten Datenübertragungen stattfinden.

Auf der Grundlage des vorliegenden Quellcodes ist nicht zu erkennen, dass Softwareumgebung oder auf einem Server ausgeführte Software mittels der Bibliothek Swashbuckle.AspNetCore während eines Einsatzes auf einem Serversystem Dokumentationsinformationen generiert und über diesen öffentlich bereitstellt.

Tatsächlich wird das vorliegende Softwareprojekt auf dem System „github“ gehostet[6] und damit grundsätzlich auf einem System Dritter in Form von Quellcode bereitgestellt. Das System github dient Entwicklern zum Austausch von Quellcode und Software und gehört zum Microsoft-Konzern[7]. Auch wenn die Projekte des Auftraggebers dort als privat markiert sind ist

[6] vgl. https://github.com/ksv-fuchs (Stand: 02.09.2023)
[7] Spiegel Online: *Microsoft kauft GitHub für 7,5 Milliarden Dollar*. 2018. Abruf: 02.09.2023, (https://www.spiegel.de/netzwelt/web/microsoft-kauft-github-fuer-7-5-milliarden-dollar-a-1211118.html)

es theoretisch aus Anwendersicht, praktisch ohne Weiteres aus Betreibersicht, möglich, Zugriff auf das Projekt und seinen Quellcode zu nehmen.

xUnit.net

Diese Bibliothek sendet erfahrungsgemäß selbst keine Daten an Dritte. Es handelt sich um ein Test-Framework, das lokal in der Entwicklungsumgebung oder in der CI/CD-Pipeline (Continuous Integration/Continuous Deployment) des Entwicklers ausgeführt wird. Sein Hauptzweck ist es, Unit-Tests für .NET-Anwendungen zu definieren, auszuführen und Ergebnisse zu berichten. Zu beachten sind allerdings:

1. **Lokale Ausführung**: xUnit.net führt Tests lokal aus, und es gibt keinen eingebauten Mechanismus, der Daten zu externen Servern sendet.
2. **Integrationen**: Während xUnit.net selbst keine Daten sendet, könnten bestimmte Integrationen oder Plugins, die Sie in Verbindung mit xUnit verwenden, dies tun. Es ist immer ratsam, die Dokumentation und Datenschutzrichtlinien von zusätzlichen Tools oder Plugins zu überprüfen, die Sie in Ihrer Testumgebung verwenden.
3. **Testdaten**: Die von Ihnen geschriebenen Tests könnten auf Daten zugreifen, die als sensibel oder vertraulich eingestuft werden. Es ist wichtig, dass solche Daten geschützt und nicht versehentlich preisgegeben werden, insbesondere wenn Sie Testberichte teilen oder Ihre Testprojekte in öffentlichen Repositories speichern.

Auf der Grundlage des vorliegenden Quellcodes ist nicht zu erkennen, dass Softwareumgebung oder auf einem Server ausgeführte Software mittels xUnit.net derart implementiert ist, dass die im Vorabsatz beschriebenen Bedingungen zur möglichen Weitergabe von möglicherweise sensiblen Daten an Dritte erfüllt sind. Daraus ist zu schließen, dass der Einsatz von xUnit.net im vorliegenden Projekt als unbedenklich einzustufen ist.

Erweiterte Quellcode-Prüfung

In der erweiterten Quellcode-Prüfung unterzieht der Autor das vorliegende Prüfobjekt einer semantischen Prüfung, um vor dem Hintergrund der Prüffrage möglicherweise bestehende Schwachstellen aufzuzeigen, die aus grundsätzlich semantisch richtigem Quellcode zur Ausführungszeit des Programms entstehen können.

FileWatcher

Im Prüfobjekt kommt ein Hintergrunddienst zum Einsatz, mit dem eine Datei-basierte Schnittstelle zu einem externen Programm ALF realisiert wird. ALF wird lt. Aussage des Auftraggebers auf demselben Produktivsystem betrieben, wie das Prüfobjekt und dient dazu, Berechnungen durchzuführen und diese in Form von Tabellenkalkulationsdateien auszugeben.

Grundsätzlich ist gegen diese Gestaltung einer Schnittstelle nichts einzuwenden. Sobald Dateien aus ALF im vom Prüfobjekt überwachten Systemverzeichnis gespeichert werden, beginnt die Verarbeitung im Prüfobjekt (vgl. *FromAlfFileWatchBackgroundService*, *ExecuteAsync*). Bei der Verarbeitung wird jedoch zunächst keine Überprüfung vorgenommen, ob die entsprechenden Dateien erwartungskonform sind. Bei einer hypothetisch angenommenen Fehlkonfiguration von Lese- und Schreibrechten auf dem Produktivsystem

wäre es daher potentiell möglich, personenbezogene Daten von Benutzern abzuändern oder zu manipulieren.

WooCommerce-Integration

Das vorliegende Prüfobjekt nutzt die Schnittstelle von WooCommerce, um Shop-Funktionen in das Programm zu integrieren. Der Auftraggeber betreibt über seine mit dem Programm WordPress gehostete Webseite einen Online-Shop8, aus dem per WooCommerce API beispielsweise Bestellinformationen durch den Quellcode des Prüfobjekts abgerufen werden (vgl. *Order.cs*, Zeile 6). Diese enthalten neben Bestelldaten personenbezogene Daten.

Die WooCommerce API, insbesondere die REST API[9], wurde nach Einschätzung des Autors auf aktuellem Stand der Technik entwickelt. Diese ist als sicher einzustufen, vorausgesetzt, sie wird korrekt konfiguriert und verwendet. Allerdings hängt die Sicherheit nicht nur von WooCommerce selbst ab, sondern auch von der Konfiguration der die API aufrufenden Programme, dem Hosting-Umfeld und den Sicherheitspraktiken, die auf administrativer Seite angewendet werden.

Sicherheitsaspekte in WooCommerce sind:

3. **Authentifizierung**: Die WooCommerce REST API verwendet ein Authentifizierungssystem auf Basis von Consumer Keys und Secrets. Wenn diese Schlüssel sicher aufbewahrt werden und nicht kompromittiert werden, bietet die Authentifizierung ein solides Sicherheitsniveau. Hierzu bedarf es einer entsprechenden Konfiguration von Lese- und Schreibrechten und oder Datenbankrechten auf dem das Prüfobjekt ausführenden Server.
4. **Berechtigungen**: Sie können API-Schlüssel in der WooCommerce-Plattform so konfigurieren, dass sie nur Lesezugriff oder Lese-/Schreibzugriff haben, je nachdem, was benötigt wird. Es ist immer ratsam, das Prinzip des geringsten Privilegs anzuwenden und nur die erforderlichen Berechtigungen zu gewähren.
5. **Transport Layer Security (TLS)[10]**: Bei Nutzung der WooCommerce API ist es dringend empfehlenswert, ausschließlich TLS-gesicherte Verbindungen zu nutzen. Dies stellt sicher, dass alle Daten, die zwischen den Systemen übertragen werden, verschlüsselt und vor sogenannten „Man-in-the-Middle“-Angriffen geschützt sind.
6. **Softwareaktualisierungen**: Es ist entscheidend, WooCommerce stets auf dem neuesten Stand zu halten. Sicherheitslücken werden regelmäßig entdeckt und behoben, und Aktualisierungen schließen diese Lücken oft.
7. **Drittanbieter-Plugins**: Die Nutzung von Drittanbieter-Plugins stellt ein erhebliches Sicherheitsrisiko dar. Diese ist im Prüfobjekt nicht erkennbar.

Zusammenfassend lässt sich feststellen, dass die WooCommerce API sicher ist, wenn sie richtig konfiguriert und verwaltet wird. Sicherheit erfordert allerdings einen mehrschichtigen Ansatz, der die gesamte Umgebung, in der WooCommerce läuft, berücksichtigt. Es ist darüber

[8] vgl. https://ksv-fuchs.de/shop/, abgerufen am 03.09.2023.
[9] Schnittstelle, mit der Funktionen aus dem Angebot von WooCommerce durch externe Programme genutzt werden können.
[10] TLS ist besser bekannt unter dem Kürzel SSL, ein Akronym des zwischenzeitlich völlig überholten Secure Socket Layer-Prinzips, welches heute als unsicher gilt.

hinaus immer ratsam, regelmäßige Sicherheitsüberprüfungen durchzuführen und Experten-Empfehlungen hinsichtlich der Verwendung einzuhalten.

WooCommerce (hier: als WordPress-Plugin, mit dem das Prüfobjekt über die API interagiert) sammelt standardmäßig keine Daten, die direkt an die Entwickler von WooCommerce oder Automattic (das Unternehmen hinter WordPress.com und WooCommerce) gesendet werden. Allerdings gibt es einige Szenarien, in denen Daten an externe Dienste gesendet werden könnten:

- **WooCommerce Tracker**: WooCommerce verfügt über ein opt-in System namens "Tracker", welches anonymisierte Daten über die WooCommerce-Installation sammelt, um dem Entwicklungsteam Einblicke für zukünftige Entwicklungen zu geben. Dies beinhaltet Daten wie die PHP-Version, den Plugin-Status, die Anzahl der Produkte und andere nicht-personenbezogene Daten. Eine Zustimmung wird für diese Datenerfassung erbeten, und diese Funktion kann jederzeit in den WooCommerce-Einstellungen deaktiviert werden.
- **Extensions und Plugins**: Bei der Installation von Erweiterungen oder Plugins, die speziell mit externen Diensten integriert sind (z.B. Zahlungsgateways, Versanddienste, Marketing-Integrationen usw.), könnten diese Erweiterungen Daten an jene Dienste übertragen. Jede Erweiterung sollte eigene Datenschutzrichtlinien und Dokumentationen haben, die klarstellen, welche Daten übertragen werden.
- **Aktualisierungen**: Um WooCommerce auf Aktualisierungen zu überprüfen und durchzuführen, kommuniziert eine WordPress-Website (nicht nur das WooCommerce-Plugin) regelmäßig mit den WordPress.org-Servern. Diese Kommunikation enthält typischerweise technische Daten über die Website, aber lt. Herstellerangaben keine personenbezogenen oder kundenbezogenen Daten.
- **Fehlerberichterstattung und Unterstützung**: Bei Inanspruchnahme von Support-Diensten von WooCommerce oder wenn das Plugin auf Fehler stößt, können Daten zur Fehlerbehebung übermittelt werden. Dabei handelt es sich jedoch in der Regel um einen manuellen Prozess und keine automatische Datenübertragung.

Es ist generell ratsam, die offizielle Dokumentation von WooCommerce sowie die Datenschutzbestimmungen und Dokumentationen aller installierten Erweiterungen und Plugins sorgfältig zu prüfen. Eine regelmäßige Sicherheits- und Datenschutzüberprüfung der E-Commerce-Website wird ebenfalls empfohlen. Funktionsumfänge und Bedingungen können sich unangekündigt oder kurzfristig ändern, auch ohne Benachrichtigung.

Applikationseinstellungen in appsettings.json

Die Konfigurationsdaten der Applikation werden, wie allgemein üblich, in der Datei *appsettings.json* gespeichert. Dagegen ist nichts einzuwenden, sofern diese ausschließlich auf dem Produktivsystem, oder anderen gesicherten Systemen, gespeichert werden. Der Autor hat begründete Zweifel, dass dem so ist.

Als Teil des Prüfobjekts wurde dem Autor durch den Auftraggeber eine appsettings.json-Datei gereicht, in der tatsächliche und funktionierende Zugangsdaten und API-Schlüssel verzeichnet sind. Damit ist anzunehmen, dass diese auch anderweitig, und möglicherweise auch für Dritte zugänglich in gitHub, gespeichert und folglich kompromittiert sind. Ferner liegt nahe, dass der

zwischenzeitlich aus den Diensten des Auftraggebers entlassene Entwickler ebenfalls über Zugangsdaten zum Prüfobjekt verfügt, und damit potentiell Zugriff nehmen könnte. Werden diese Daten von den Systemen dieser möglichen Dritten entwendet, wäre der Auftraggeber möglicherweise nicht einmal in Kenntnis, dass weitere Dritte sich völlig unbemerkt Zugang zu seinen Systemen und den dort hinterlegten personenbezogenen Daten seiner Kunden verschaffen könnten.

Den Applikationseinstellungen ist unmittelbar der Zugriff des zwischenzeitlich aus den Diensten des Auftraggebers entlassenen Entwicklers zu entnehmen, der dort Diagnose- und personenbezogene Daten erhält (s. „Sentry“, *appsettings.json*, Zeile 46ff.), außerdem als Standard-Administrator hinterlegt ist (vgl. *appsettings.json*, Zeile 16f.).

Bei der Registrierung eines neuen Benutzers ist es erforderlich, dass dieser die Registrierung per Aktivierungs-Link aus einer (per SendinBlue) versendeten E-Mail bestätigt und damit sein beim Auftraggeber im Prüfobjekt geführtes Konto aktiviert. Die Implementierung eines Standard-Passworts für noch nicht aktivierte Konten (vgl. *appsettings.json*, Zeile 7) kann zu schwerwiegenden Sicherheitsrisiken führen, falls ein Benutzer eine Aktivierung durchgeführt hat, jedoch kein neues Passwort gesetzt hat, und in diesem Falle das in der Konfiguration hinterlegte Standardpasswort gilt.

Paketquellen nuget.config

Mittels der Entwicklungsumgebung Microsoft Visual Studio lassen sich externe Programm-Bibliotheken aus öffentlich zugänglichen Quellen laden. Hierzu betreibt Microsoft das Portal NuGet, welches einen Zugang zu öffentlichen Programm-Bibliotheken, zu über 90% aus Open-Source-Projekten bereitstellt. Bei Open-Source-Projekten lässt sich nicht nur die Herkunft eindeutig bestimmen, sondern ebenfalls der Quellcode und seine Funktionsweise einsehen. Dadurch wird es auch möglich, Unbedenklichkeit oder mögliche Implikationen hinsichtlich des Datenschutzes festzustellen. In Abhängigkeit der Lizenzierung lassen sich diese über NuGEt bezogenen Bibliotheken in neue Programme integrieren.

Die im Rahmen dieses Dokuments nicht aufgelisteten Bibliotheken stammen aus sicheren Quellen und sind nach aktuellem Stand der Technik und dem Erfahrungsstand des Autors unbedenklich.

Die Paketkonfigurationsdatei des Projekts enthält (neben dem sicheren NuGet) externe Quellen, die dem Auftraggeber potentiell nicht länger zur Verfügung stehen könnten. Hierbei handelt es sich mit der Quelle „BCX“ (vgl. *nuget.config*, Zeile 4) offenbar um eine vom zwischenzeitlich nicht mehr durch den Auftraggeber beschäftigten Entwickler betriebene Quelle, der den Zugang möglicherweise begrenzen kann (vgl. *nuget.config*, Zeile 8ff.), indem er die Zugangsdaten ändert oder sperrt. Im Ergebnis wäre das Prüfobjekt für den Auftraggeber nur noch in dem Zustande verwendbar, indem es auf dem Produktivsystem aufgespielt wäre, eine Instanthaltung, Wartung, Weiterentwicklung (u.a. durch Dritte) wäre dann nicht oder nur mit erheblichem Aufwand möglich[11].

[11] Hier müsste man sämtliche aus den Bibliotheken mit dem Präfix BCX genutzten Funktionalitäten durch andere Bibliotheken ersetzen oder selbsttätig implementieren. Der Aufwand wäre erheblich.

Der Betreiber der Paketliste „BCX“ erhält potentiell die Möglichkeit, mittels einer Zugriffstatistik der von ihm gehosteten Bibliotheken festzustellen, ob, wann, in welchem Umfang, wo und ggf. von wem der Auftraggeber eine Bereitstellung des Prüfgegenstandes auf einem Produktivsystem, Instanthaltung, Weiterentwicklung, etc. vornehmen lässt. Die Namen der Pakete lassen darauf schließen, dass es sich hierbei um Kernfunktionalitäten handeln kann, die lediglich statt vom Hersteller (z.B. Microsoft) vom Betreiber „BCX“ bereitgestellt werden.

Das Vorgehen, Standard-Bibliotheken, die Microsoft ohnehin zum direkten Zugriff kostenlos bereitstellt, erneut in einer eigenen Quelle bereitzustellen, ist grundsätzlich unnötig (vgl. *KsvShopOrdersController.cs*: BCXF.AspNetCore.Core.Attributes als ursprünglicher Bestandteil von ASP.Net Core[12]). Es sind allerdings noch weitere Bibliotheken eingebunden, wie etwa und rein exemplarisch, und nicht darauf beschränkt, in *KsvShopOrdersController.cs,* BCXF.Extension.AspNetCore.Mvc, BCXF.Extension.NetCore, BCXF.Extension.NetCore.Collections, die der Namensgebung nach Kernfunktionalitäten des ursprünglichen Frameworks erweitern. Der Quellcode ist hier allerdings nicht öffentlich zugänglich. Hier wäre gesondert zu prüfen, in welchem Umfang und auf welche Art eine solche Erweiterung stattfindet, da es für den Autor ohne Vorliegen des Quellcodes dieser Bibliotheken, oder aber einen Auftrag diese in einem Reverse-Engineering-Verfahren zu öffnen und zu untersuchen, nicht möglich ist, hier eine Aussage zur Prüffrage zu treffen.

Im Kontext der DSGVO geht es vor allem darum zu verhindern, dass personenbezogene Daten an Dritte weitergegeben werden oder dass unsichere Bibliotheken das Risiko von Datenverletzungen erhöhen. Es empfiehlt sich daher folgendes Vorgehen:

1. **Vertrauenswürdige Quellen nutzen**: Bekannte und vertrauenswürdige Quellen wie etwa NuGet[13] für den Bezug von Bibliotheken verwenden. Den Einsatz von Bibliotheken aus unsicheren oder unbekannten Quellen vermeiden.
2. **Lizenzen überprüfen**: Sicherstellen, dass jede verwendete Bibliothek über eine klare Lizenz verfügt. Einige Lizenzen können Einschränkungen oder Anforderungen enthalten, die den Datenschutz betreffen.
3. **Dependency Scanning**: Tools verwenden, die Abhängigkeiten in Softwareprojekten analysieren, um bekannte Sicherheitslücken oder Probleme zu identifizieren.
4. **DSGVO-Relevanz**: Prüfen, ob die Bibliothek mit personenbezogenen Daten arbeitet und oder Daten an externe Dienste sendet.
5. **Dokumentation & Community**: Die offizielle Dokumentation der Bibliothek und oder ihren Quellcode überprüfen. Eine aktive Community und gut gepflegte Dokumentation sind oft Indikatoren für vertrauenswürdige Bibliotheken.
6. **Code Review**: Wenn möglich, den Quellcode der Bibliothek überprüfen.
7. **Datentransfer**: Monitoring-Tools verwenden, um den Datenverkehr der Anwendung zu überwachen und unerwartete Datenübertragungen zu erkennen.
8. **Datenverarbeitungsverträge (DPAs)**: Falls eine Bibliothek oder ein Dienst Daten zu externen Parteien sendet, notwendige Datenverarbeitungsverträge mit diesen Anbietern abschließen.

[12] Microsoft: *Create web APIs with ASP.NET Core*. 11.04.2023. Abgerufen am: 04.09.2023 (https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-7.0)
[13] vgl.: https://www.nuget.org

9. **Regelmäßige Aktualisierungen durchführen**: Bibliotheken regelmäßig aktualisieren, um bekannte Sicherheitslücken zu beheben.

Bewertung und Interpretation

Der Befund zeigt deutliche Schwachstellen im Prüfobjekt, die die Datensicherheit und die Einhaltung datenschutzrechtlicher Vorschriften generell oder unter spezifischen Umständen gefährden. Die Eintrittswahrscheinlichkeit der spezifischen Umstände werden dadurch begünstigt, dass keine geeignete Dokumentation vorliegt, die einem Administrator ohne Kenntnisse des Quellcodes eine ordnungsgemäße Installation der Software für einen Produktivbetrieb auf einem Server ermöglicht.

Konkret sind folgende Punkte als kritisch zu bewerten:

- Eine mangelhafte Konfiguration eines Produktivsystems, auf dem das Prüfobjekt zur Nutzung für Benutzer über das Internet bereitgestellt wird, kann scherwiegende Folgen haben, und ist insbesondere hinsichtlich der Schreib- und Leserechte als kritisch zu bewerten. Die ClosedXML-Schnittstelle kann beispielsweise potentiell zur Platzierung von schadhaftem Code auf dem System verwendet werden, sofern der Server nicht mit für den spezifischen Anwendungsfall der Ausführung des Prüfobjekts konfigurierte Lese- und Schreibrechte konfiguriert wurde. Auch wenn grundsätzlich nichts gegen die Verwendung der externen Bibliothek ClosedXML spricht, muss im Produktivsystem zwingend eine Trennung zwischen Bereitstellungs- und Datenverzeichnissen vorgenommen werden.

- Die im Prüfobjekt vorgenommene Integration von WooCommerce (oder Daten aus WooCommerce) ist nur dann als unbedenklich einzustufen, wenn auch die jeweils korrespondierend verwendete Shop-Instanz der WooCommerce API nach Stand der Technik konfiguriert und eingesetzt wird.

  Die hier bezüglich WooCommerce festgestellten Sicherheits- oder datenschutzrelevanten Aspekte (s.o.) verstehen sich rein indikativ, erheben keinen Anspruch auf Vollständigkeit und sind als minimale Notwendigkeit in einem Produktivbetrieb zu betrachten. Je nach Konfiguration, Servereinsatz, Produktivszenario und ggf. weiteren, hier nicht abschließend nennbaren Faktoren, sind weitere Konfigurationen, Absicherungen, zusätzliche Einstellungen usw. notwendig, auch wenn das Prüfobjekt an sich in dieser Hinsicht nach Auffassung des Autors keiner konkreten Veränderung bedürfte.

- Die Nutzung der Bibliothek Sentry.AspNetCore im Prüfgegenstand ist hochgradig bedenklich, da dort Daten über die Sentry-API an Dritte weitergegeben werden, die neben (möglicherweise) anonymisierten, rein statistischen Daten im Prüfgegenstand insbesondere personenbezogene Daten enthalten. Es ist nicht erkennbar, dass hierzu eine Notwendigkeit besteht, vielmehr werden dort Daten auf Systeme des vormaligen Entwicklers übertragen, zu dem lt. hier vorliegender Auskunft des Auftraggebers kein Kontakt mehr besteht.

- Die Nutzung von SendinBlue ist hinsichtlich des Versands von Benutzer-, Konto-, Aktivierungs- oder Zugangsdaten über diesen Dritten als potentiell bedenklich einzustufen. Der Versand von weiteren personenbezogenen Daten ist sowohl als

hochgradig bedenklich, wie auch unnötig, einzustufen. In diesem Zusammenhang liegt dem Autor zum Zeitpunkt der Erstellung dieser Analyse keine Auftragsverarbeitungsvereinbarung des Auftraggebers mit SendinBlue vor.

- Die Nutzung von gitHub als Repository-System eines Nicht-Open-Source-Projekts stellt ein potentielles Risiko dar, da der Quellcode vollständig durch Dritte – auch vom Auftraggeber möglicherweise völlig unbemerkt – eingesehen oder abgegriffen werden kann. Insofern ist auch die hier rein hypothetisch anzunehmende Hinterlegung von API-Zugangsdaten im Repository besonders kritisch[14].

- Der im Prüfobjekt praktizierte Umgang mit Zugangsdaten und API-Schlüsseln ist als hochgradig bedenklich einzustufen. Folglich müssen alle Zugangsdaten verändert und API-Schlüssel neu generiert werden, sofern die dazugehörigen API denn genutzt werden sollen.

- Die Abhängigkeit des Prüfobjekts, auch im Hinblick auf Wartbarkeit, Instanthaltung und Weiterentwicklung, von den Programm-Bibliotheken mit dem Präfix „BCX“ ist als kritisch zu beurteilen.

- Die Überprüfung der Herkunft und Integrität von Programm- oder Softwarebibliotheken in einem C#-Projekt (oder in jedem Softwareprojekt) ist aus verschiedenen Gründen wichtig, einschließlich Sicherheit und Datenschutz, insbesondere im Kontext der Datenschutz-Grundverordnung (DSGVO). Die Programm-Bibliotheken mit dem Präfix „BCX“, die Erweiterungen zu anderen Programm-Bibliotheken darstellen und deren Quellcode nicht öffentlich zugänglich ist, konnten vom Autor nicht geprüft werden und sind als kritisch einzustufen.

[14] Der Autor hat die Erfahrung gemacht, dass Entwickler Konfigurationsdaten ebenfalls in Repository-Systeme übertragen. Dies stellt eklatante Sicherheitsrisiken dar, da Konfigurationsdaten, wie auch in dem Prüfgegenstand, (API)-Zugangsinformationen zu Systemen enthalten können, die keinesfalls an Dritte weitergegeben werden dürfen. Die Prüfung des gitHub-Repositories war nicht Gegenstand dieses Auftrags.

Kurzanalysebericht Nr. NA08/23
zur Server-Software „KSV-Frontend“ und „KSV-Backend“

von David Bouck-Standen, M.Sc., Lilienstr. 11, 20095 Hamburg
Auftraggeber: KSV-Fuchs, Herr Frank Fuchs, Ludwig-Erhard-Allee 10, 76131 Karlsruhe

Einleitung
Der vorliegende Analysebericht zur Software wurde sorgfältig von David Bouck-Standen erstellt, der mit mehr als zwei Jahrzehnten Erfahrung auf dem Gebiet der Informationstechnologie und qualifizierten Abschlüssen B.Sc. und M.Sc. einer deutschen Universität in Informatik nach mehrjähriger wissenschaftlicher Tätigkeit inzwischen als Senior Software-Systems-Engineer, außerdem als Experte in der IT-Sicherheit tätig ist.

Beschreibung des Prüfobjekts
Der Prüfgegenstand besteht aus den in zwei komprimierten ZIP-Dateien vorgelegten Software-Projekten mit dem Titel „KSV-Frontend“ und „KSV-Backend“. Die Software-Projekte enthalten Programmcode, auch Quellcode genannt, aus dem die Softwareprogramme bestehen. Das Projekt „KSV-Frontend“, im Folgenden als „Frontend“ bezeichnet, realisiert einen Teil der für den Auftraggeber entwickelten Softwareanwendung, den der Endbenutzer direkt verwenden und sehen kann, z.B. über die Benutzeroberfläche. Diese wird im Web-Browser (z.B. Safari, Firefox, Edge) angezeigt und zu diesem Zwecke von einem Server ausgeliefert. Auf diesem Server muss das Frontend betrieben werden. Zur vollen Funktionsfähigkeit wird außerdem das im Folgenden als „Backend“ bezeichnete Software-Programm aus dem Projekt „KSV-Backend“ benötigt. Das Backend bezieht sich auf den serverseitigen Teil der für den Auftraggeber entwickelten Softwareanwendung, der für Datenverarbeitung, Speicherung und Serverlogik verantwortlich ist. Es ist nicht direkt für den Endbenutzer sichtbar. Die vom Endbenutzer vorgenommenen Eingaben in das Frontend werden durch dieses an das Backend weitergeleitet. Die Systemantwort aus der Programmlogik leitet das Backend an das Frontend weiter, in dem es für den Endbenutzer angezeigt wird.

Zielsetzung
Diese Kurzanalyse geht der folgenden Fragestellung nach:

- Wurde der Programmcode an Dritte weitergegeben?

Methodik
Für diese Analyse wird der vorliegende Programmcode und insbesondere die Projektdateien, Zeile für Zeile untersucht mit dem Ziel, eine Aussage hinsichtlich der Prüffrage zu treffen. Projektdateien enthalten Informationen zu Projekten, Projektabhängigkeiten und Metadaten, die von der jeweiligen Entwicklungsumgebung genutzt werden, in der das Programm programmiert wird.

Befund
Der Quellcode weist hinsichtlich der Fragestellung relevante Code- oder Einstellungszeilen auf. Die Unterabschnitte dieses Abschnitts werden nach den entsprechenden Dateinamen bezeichnet, auf die sich die Unterabschnitte beziehen.

Dockerfile
Das Dockerfile beschreibt den Bau und die Laufzeitumgebung einer .NET Webanwendung. Es verwendet einen Multistage-Bau, bei dem die Anwendung zunächst im Bauabschnitt gebaut und dann im Laufzeit-Container ausgeführt wird. Darüber hinaus installiert es zusätzliche Tools und Schriftarten im Laufzeit-Container.

Im Backend weist das Dockerfile folgende Codezeilen (18f.) auf:
```
### Github Specific
LABEL org.opencontainers.image.source https://github.com/KSV-Fuchs/Webapp_Backend
```

Im Frontend weist das Dockerfile folgende Codezeilen (18f.) auf:
```
### Github Specific
LABEL org.opencontainers.image.source https://github.com/KSV-Fuchs/Webapp_Frontend
```

Hierbei handelt es sich um eine für die Online-Plattform „GitHub“ spezifische Anweisung: Diese fügt dem Container-Bild (Image) eine Bezeichnung (Label) hinzu, die die Quelle des Codes angibt. In diesem Fall ist es ein GitHub-Repository.

GitHub ist eine Plattform für Softwareentwicklung und -versionierung, die Git als zentralen Versionierungsdienst verwendet. Git ist ein verteiltes Versionskontrollsystem, das es Entwicklern ermöglicht, Quellcode-Änderungen zu verfolgen und mit anderen zusammenzuarbeiten. GitHub wurde 2008 gegründet und hat sich seitdem zu einer der bekanntesten und meistgenutzten Plattformen für Softwareentwicklung und Open-Source-Projekte entwickelt. Es wurde 2018 von Microsoft erworben.

Das unter dem Link https://github.com/KSV-Fuchs/Webapp_Backend oder https://github.com/KSV-Fuchs/Webapp_Frontend bei GitHub gehostete Repository ist als privat markiert. Das bedeutet, dass GitHub den Inhalt nur vom Ersteller des Repositories (und des Benutzerprofils, von welchem das Repository angelegt wurde) zugelassenen Benutzern und dem Ersteller selbst anzeigt. Wird ein GitHub-Repository betrieben bedeutet dies, dass alle darin enthaltenen Daten, d.h. alle Quellcode-Dateien und damit auch sämtlicher Quellcode, beim Anbieter GitHub gespeichert werden.

Im voranstehend bezeichneten Dockerfile findet sich des Weiteren die Codezeile (Z. 15):
```
RUN wget https://gist.githubusercontent.com/mwimmer-
bcx/1cdf7660b6e91bc37ef4e7dce9422639/raw/1f93b82aa326a75380a16dc4e84b4a5c079560ea/ttf-vista-
fonts-installer.sh -q -O - | bash
```

Die im Vorabsatz verzeichnete Codezeile lädt ein Skript von einem von GitHub bereitgestellten Server herunter, das scheinbar zur Installation von Schriftarten dient, und führt es bei der Bereitstellung der Software über das System „Docker“[1] für Betrieb der Software aus. Der abgerufene Inhalt wird hierbei über GitHub von einem Endbenutzer mit dem Benutzernamen „mwimmer-bcx“ bereitgestellt, der auf seiner Profilseite[2] seine personenbezogenen Daten über

[1] Docker ist eine Plattform, die es Entwicklern und Systemadministratoren ermöglicht, Anwendungen in Containern zu erstellen, auszuführen und zu verteilen. Ein Container ist eine standardisierte Einheit, die Code, Laufzeitmodul, Systembibliotheken und Systemeinstellungen beinhaltet. (vgl. https://docker.com, abgerufen am: 10.09.2023).
[2] Siehe https://gist.githubusercontent.com/mwimmer-bcx

GitHub veröffentlicht, darunter seinen Namen mit „Manuel Wimmer“, sein Foto und seine Betriebszugehörigkeit mit „@BroadcastX-GmbH“ angibt.

Bewertung und Interpretation

Die Konfigurationsdateien beider Projekte, Frontend wie Backend, geben begründeten Anlass zu der Annahme, dass der Quellcode beider Projekte vollständig bei GitHub hinterlegt wurden. Diese Annahme bestätigt sich durch Aufrufen des Weblinks https://github.com/KSV-Fuchs/, der zu dem GitHub-Benutzerprofil mit dem Namen des Auftraggebers „KSV-Fuchs“ führt. Dort wird auch das Logo des Auftraggebers geführt. Der Autor erhielt nach Rücksprache mit dem Auftraggeber die Auskunft von diesem, dass dieser in Unkenntnis der Existenz eines (für ihn) eingerichteten GitHub-Profils sei, ferner weder in Kenntnis von einem Hochladen seines Quellcodes zu GitHub sei, noch diesem jemals zugestimmt hätte.

Die beiden Repositories für Frontend und Backend sind als „privat“ eingestellt, d.h. nur vom Ersteller (des Profils und der Repositories für Frontend und Backend) festgelegte Benutzer können darauf mit von GitHub bereitgestellten Methoden zugreifen[3].

Der Ersteller hat mit dem Hochladen des Quellcodes gem. der dortigen Nutzungsbedingungen „The GitHub Terms of Service“ dem Anbieter GitHub und sämtlichen Rechtsnachfolgern eine zeitlich unbegrenzte Lizenz zur Speicherung („store“, „make copies“), Archivierung („archive“), elektronischen Verarbeitung („parse“), Veröffentlichung („display“) und einer nicht näher spezifizierten elektronischen Analyse („otherwise analyze“) erteilt[4]. Weitere (juristische) Implikationen ergeben sich möglicherweise, jedoch nicht abschließend aufgezählt, aus den weiteren Nutzungsbedingungen des Anbieters GitHub, den Vereinbarungen zwischen dem den Code erstellenden Auftragnehmer und dem Auftraggeber des Autors, außerdem weiteren Normen und Verordnungen.

[3] Originaltext: „GitHub considers the contents of private repositories to be confidential to you. GitHub will protect the contents of private repositories from unauthorized use, access, or disclosure in the same manner that we would use to protect our own confidential information of a similar nature and in no event with less than a reasonable degree of care.“ (vgl. „The GitHub Terms of Service“, Abschnitt E. 2.) Quelle: https://docs.github.com/en/site-policy/github-terms/github-terms-of-service#the-github-terms-of-service (abgerufen am 10.09.2023).

[4] Originaltext: „We need the legal right to do things like host Your Content, publish it, and share it. You grant us and our legal successors the right to store, archive, parse, and display Your Content, and make incidental copies, as necessary to provide the Service, including improving the Service over time. This license includes the right to do things like copy it to our database and make backups; show it to you and other users; parse it into a search index or otherwise analyze it on our servers; share it with other users; and perform it, in case Your Content is something like music or video.

This license does not grant GitHub the right to sell Your Content. It also does not grant GitHub the right to otherwise distribute or use Your Content outside of our provision of the Service, except that as part of the right to archive Your Content, GitHub may permit our partners to store and archive Your Content in public repositories in connection with the GitHub Arctic Code Vault and GitHub Archive Program.“ “The GitHub Terms of Service”, D. 4. Abgerufen am 10.09.2023, Quelle: https://docs.github.com/en/site-policy/github-terms/github-terms-of-service#4-license-grant-to-us

Dem Autor ist es mangels eines Zugangs zu den vormals erwähnten Repositories nicht möglich zu prüfen, ob und falls ja, welcher Quellcode dort hochgeladen wurde. Der dem Unterzeichnenden vorliegende Quellcode-Export in Form der ZIP-Dateien, ebenfalls auch deren GitHub-spezifische Datei-Benennung[5], legen nahe, dass der Quellcode beider Projekte, Frontend und Backend, vollständig bei GitHub vorliegt. Es ist technisch außerdem möglich, dass der dort hochgeladene Quellcode weitere Binärdaten enthält, wie etwa das in den Testprojekten referenzierte „ALF“ oder „ZinsID“, welches der Auftraggeber entwickeln lies und insbesondere zur Funktionsfähigkeit des hier im Prüfgegenstand vorliegenden Backends benötigt wird. Auch wenn Daten zwischenzeitlich aus einem aktuellen Stand von GitHub entfernt wurden bedeutet dies, dass diese tatsächlich noch in der „History“, der Verlaufshistorie der Programmierung, liegen, was eine zentrale Funktionalität des Portals GitHub darstellt.

Die Bereitstellung von Quellcode auf GitHub, sei es in öffentlichen oder privaten Repositories, stellt gem. den dortigen Bedingungen durch den Akt des Hochladens eine Weitergabe des Quellcodes an Dritte dar. GitHub ist z.B. zu einer nicht näher spezifizierten elektronischen Analyse des Prüfobjekts berechtigt. Es kann im Rahmen dieses Berichts auch nicht abschließend geklärt werden, welcher Art, Umfang oder Dauer die Analyse stattfindet. Es ist ferner für den Autor nicht ersichtlich, auf welchen und wie vielen ggf. durch weitere Dritte betriebenen Servern sich das Prüfobjekt des Auftraggebers nun befindet, und ob und in welchem Umfang ein Zugriff ggf. durch weitere Dritte möglich ist. Dies ist als äußerst kritisch i.S.d. Prüffrage zu bewerten.

Der Quellcode des Auftraggebers befindet sich nun potentiell auf einer nicht näher spezifizierten Anzahl an Systemen Dritter, die außerhalb des Kenntnis-, und Verantwortungsbereichs sowie Zugriffs des Auftraggebers liegen. Sofern hier beispielsweise ein unberechtigter Zugriff eines Dritten (sei es intern bei GitHub, einem von GitHub eingesetzten Subunternehmer, oder eines „Hackers“, der sich unberechtigterweise Zugriff auf den Code bei GitHub oder Subunternehmern von GitHub nimmt, oder möglicherweise durch einen Zugriff auf Systeme des Entwicklers des Auftraggebers sich des Codes bemächtigt), wäre nicht sichergestellt, dass der Auftraggeber überhaupt Kenntnis dieses Zugriffs erhalten würde, und diesen auch nicht verhindern könnte.

Bei der Frage nach dem Ersteller des GitHub-Repositories, das unter dem Profilnamen des Auftraggebers geführt wird (s.o.), legt der ebenfalls im Quellcode hinterlegte direkte Verweis auf den bei GitHub gehosteten User-Content (s.o.) sowie die Auskunft des Auftraggebers, bei dem betreffenden Benutzer handele es sich um den die Software erstellenden Softwareentwickler, nahe, dass dieser Benutzer das entspreche Repository erstellt haben könnte. Der Autor empfiehlt hier dringend die Einholung einer verbindlichen Auskunft des Betreibers GitHub. Ferner sollte auch dringend die Auskunft eingeholt werden, welche Personen sonst noch Zugriff auf den Quellcode des Auftragnehmers nehmen konnten oder noch können.

[5] Jedes Repository lässt sich mit GitHub-Funktionen exportieren als ZIP-Datei mit gewissem Layout, typischerweise wird dies Benannt mit <Name des Repositories>-<Branch-Name>.zip – dies ist hier bei Front- und Backend-Projekt jeweils der Fall.

Die Nutzung von GitHub für Projekt, das nicht als Quelloffenes Open-Source-Projekt bereitgestellt werden soll, sondern spezifisch als Individualsoftware für einen spezifischen Auftraggeber entwickelt wurde, als kritisch. Das aus u.a. den USA gehostete GitHub (Sitz des Betreibers[6]) unterliegt nicht nur den dortigen Nutzungsbedingungen, sondern diese ebenfalls einem einem anderen Rechtsraum entspringenden Gesetzen und Verordnungen, die für den Auftraggeber weder verständlich noch greifbar sein können.

Die Wahl von GitHub als ein Repository-System für ein dem Geschäftsgeheimnis unterliegendes Softwareprojekt bewertet der Autor als kritisch und verantwortungslos gegenüber dem Auftraggeber. Stattdessen hätte beispielsweise das auf einem eigenen Entwicklungsserver (oder System des Auftraggebers) betriebene „GitTea“[7] eingesetzt werden können, dass über einen Funktionsumfang analog GitHub verfügt, jedoch jederzeit die vollständige Kontrolle über den Quellcode oder einen Zugriff auf den Quellcode ermöglicht. Nach Auffassung des Autors hätte der Einsatz von GitHub in dem vorliegenden Prüfobjekt die Zustimmung des Auftraggebers erforderlich gemacht, der zuvor über mögliche Risiken hätte aufgeklärt werden müssen.

Zusammenfassung
Das vorliegende Prüfobjekt mit seinen zwei Bestandteilen, Frontend- und Backend-Software, enthält stichhalte Hinweise darauf, dass der Quellcode auf das System mindestens eines Dritten übertragen wurde. Für den Autor stellt sich die Übertragung dar, als sei diese ohne das Wissen und Wollen des Auftraggebers geschehen.

Sofern die im Befund dieses Berichts festgestellten Quellcodeübertragungen an den Dritten, hier GitHub, stattgefunden haben, wurde diesem Dritten gem. dessen Bedingungen durch das Hochladen eine noch näher juristisch zu bewertende Nutzungslizenz am Prüfobjekt des Auftraggebers eingeräumt, die dieser Dritte ebenfalls in Form der Unterlizenzierung auf weitere Dritte zu übertragen berechtigt ist. Dies geschah gem. den Ausführungen des Auftraggebers ohne dessen Wissen und Wollen. Dem Autor (und dem Auftraggeber) stehen mit Datum dieses Berichts keine Möglichkeiten zur Verfügung,

(a) Zugriff auf das auf den Namen des Auftraggebers erstellte GitHub-Profil zu nehmen und
(b) die dort hochgeladenen Repositories für Frontend- und Backend-Projekt einzusehen (oder zu verwalten und z.B. von GitHub zu löschen).

Der Autor sieht einen dringenden Auskunftsbedarf seitens GitHub,

(i) ob und welche Daten dorthin übertragen wurden, und ferner
(ii) wer die Daten übertragen und
(iii) dazu berechtigt wurde, diese einzusehen, zu verändern oder herunterzuladen.

Hamburg, den 10.09.2023

David Bouck-Standen, M.Sc.

[6] Sitz der Betreibergesellschaft GitHub Inc. ist San Francisco, Kalifornien, USA (vgl. https://github.com/about).
[7] vgl. GitTea, https://gittea.dev

Anlagen

Quellcode des Prüfobjekts.

Zusammenfassung

Das Prüfobjekt kann in seinem aktuellen Zustand nicht ohne Bedenken betrieben werden, da das Prüfobjekt aufgrund des hier festgestellten Befunds kritische datenschutzrechtlich relevante Aspekte aufweist, die die Datenintegrität und Vertraulichkeit i.S.d. DSGVO verletzen.

Während des Betriebs des Prüfobjekts werden personenbezogene Daten von Benutzern in mindestens drei voneinander unabhängigen Fällen an Dritte übertragen. Auch wenn dies, wie im Falle von SendinBlue offenbar z.B. hinsichtlich des Versands von Zugangsdaten, vermutlich beabsichtigt war, sind die Datenübertragungen in jedem Falle vermeidbar. Im Falle der Übermittlung von Daten an die Internetadresse „sentry.broadcastx.io“, die (empfindliche) personenbezogene Daten in erheblichem Umfang enthalten, ist diese Datenübertragung völlig unzulässig.

Mangels vorliegender Dokumentation oder Quellcode zu den externen Programmbibliotheken mit Präfix „BCX“ muss zur Wahrung der Datenintegrität und Datensicherheit davon ausgegangen werden, dass hier möglicherweise personenbezogene Daten verarbeitet oder an Dritte gesendet werden. Eine Reputation, etwa wie bei NuGet, ist außerdem nicht vorhanden. Daher ist es dringend erforderlich, die Programmbibliotheken auszutauschen.

Das Prüfobjekt bedarf für einen sicheren Betrieb eines ordnungsgemäß konfigurierten Serversystems. Allerdings sind die dafür notwendigen Parameter für den Autor nirgendwo ersichtlich dokumentiert.

Auch zur Sicherung der Instandhaltung, Wartbarkeit und Weiterentwicklung bedarf es zur Vermeidung von Abhängigkeiten von Dritten mittel- und langfristig des Austauschs der mit dem Präfix „BCX“ versehenen Programm-Bibliotheken, auch wenn für diese eine Dokumentation gereicht werden sollte.

Hamburg, den 04.09.2023

David Bouck-Standen, M.Sc.