Swiss Covid Certificate - Public Security Test

Aktuelle Meldungen - Notifications actuelles - Notifiche attuali - Current reports

28.06.2021 / 08:00

Number	Subject	Date received	Summary / Keyword	Description	Impact	Mitigation	Credits	Status	Remarks
1	Cert Error	31.5.2021		Cert Error for the website covidcertificate.admin.ch	Website certificate is not trusted by browser	Message to BK sent (31.05.2021)		Analysis	
2	Signatur von beliebigen Eingaben	31.5.2021		https://github.com/admin-ch/CovidCertificate-Api-Gateway-Service/blob/develop/src/main/java/ch/admin/bag/covidcertificate/gateway/web/controller/utils/SignatureTestController.java#L41 Eine erreichbare Test-Funktionalität, welche beliebige Eingaben mit einem private key signiert und wieder zurücksendet. Eine solche Funktionalität ist grundsätzlich unsicher, beliebige Inhalte ohne zusätzliche Überprüfung zu signieren ist ein Problem, schliesslich sollte immer geprüft werden ob etwas signiert werden soll.	Mit hoher Warscheinlichkeit keine Auswirkung bezüglich technischer Security, da der hartkodierte Pfad des Private Keys im Release hoffentlich nicht besteht, der Code niemals die Produktion erreicht und ähnliche Vorbedingungen. Auswirkung grundsätzlich daher unklar da lediglich anhand von code review gefunden. Code stört jedoch bei Code review da es sich um ein klares kryptografisches Anti-Pattern handelt.	Grundsätzlich sollte eine solche Klasse nicht im *main* Ordner vorhanden sein oder überhaupt im produktiven Code, wenn dann höchstens im Test-Ordner, aber auch da lieber nicht.		Won't fix	Not in PROD deployment.
3	Supply Chain attack	31.5.2021	Supply Chain attack possible due to untrusted third-party GitHub Actions usage	Multiple components make use of external untrusted third-party GitHub Actions such as these ones: - https://github.com/marvinpinto/action-automatic-releases - https://github.com/ncipollo/release-action The following workflows make use of these untrusted actions: - `admin-ch/CovidCertificate-Management-Service/.github/workflows/release.yml` - `admin-ch/CovidCertificate-Api-Gateway-Service/.github/workflows/release.yml` - `admin-ch/CovidCertificate-App-Config-Service/.github/workflows/tagged_release.yaml` - `admin-ch/CovidCertificate-Api-Gateway-Service/.github/workflows/maven.yml` - `admin-ch/CovidCode-Service/.github/workflows/maven.yml` - `admin-ch/CovidCertificate-Management-Service/.github/workflows/maven.yml` - `admin-ch/CovidCertificate-App-Config-Service/.github/workflows/maven.yml`	As per GitHub's Official Documentation (https://docs.github.com/en/actions/learn-github-actions/security-hardening-for-github-actions#using-third-party-actions): ``` The individual jobs in a workflow can interact with (and compromise) other jobs. ... This means that **a compromise of a single action within a workflow can be very significant**, as that compromised action would have access to all secrets configured on your repository, and may be able to use the `GITHUB_TOKEN` to write to the repository. ``` In a worst case scenario: both users `marvinpinto` and `ncipollo` could release malicious code that would fully compromise the integrity of `admin-ch`'s supply chain and (by extension) software builds. Recently such attacks were used against high-profile targets such as Codecov and Solarwinds.	You can help mitigate this risk by following these good practices (As per GitHub's Documentation): ### Pin actions to a full length commit SHA Pinning an action to a full length commit SHA is currently the only way to use an action as an immutable release. Pinning to a particular SHA helps mitigate the risk of a bad actor adding a backdoor to the action's repository, as they would need to generate a SHA-1 collision for a valid Git object payload. ### Audit the source code of the action Ensure that the action is handling the content of your repository and secrets as expected. For example, check that secrets are not sent to unintended hosts, or are not inadvertently logged. ### Pin actions to a tag only if you trust the creator Although pinning to a commit SHA is the most secure option, specifying a tag is more convenient and is widely used. If you'd like to specify a tag, then be sure that you trust the action's creators. The		Analysis	

				The following link shows that `marvinpinto/action-automatic-releases` was executed while building the latest `master` release: https://github.com/admin-ch/CovidCertificate-Management-Service/runs/2708296617?check_suite_focus=true	The following article provides an interesting insight on these ones: https://www.wolfe.id.au/2021/04/26/github-actions-supply-chain-attacks/	'Verified creator' badge on GitHub Marketplace is a useful signal, as it indicates that the action was written by a team whose identity has been verified by GitHub. Note that there is risk to this approach even if you trust the author, because a tag can be moved or deleted if a bad actor gains access to the repository storing the action.			
4	Accessibility	01.06.2021	Anforderungen WCAG 2.1	Dies ist eine generelle Anforderung. Damit Menschen mit Behinderungen die App bedienen können, müssen die Anforderungen WCAG 2.1 der Konformitätsstufe AA erfüllt sein. Dies gilt auch für native Apps für Android und iOS. Die ist eine zwingende Vorgabe gemäss Behindertengleichstellungsgesetzt (BehiG). Weitere Informationen im eCH-Standard 0059. Es braucht hier detaillierte Accessibiltiy-Audits.	Falls die App nicht barrierefrei bedient werden kann, ist die besonders gefährdete Gruppe von Menschen mit Behinderungen ausgeschlossen. Die gesetzlichen Vorgaben sind nicht eingehalten. Gefahr der Ausgrenzung von Menschen mit Behinderungen und älteren Menschen und von Image- und Reputationsverlust.	Vollständige Accessibility Audits und kontinuierliche Verbesserung der Mängel.		Analysis	
5	Webserver	02.06.2021		Your web server supports one or more ciphers that have a phase out status, because they are known to be fragile and are at risk of becoming insufficiently secure.	Technical details: Web server IP address Affected ciphers Status 162.23.147.222 AES256-GCM-SHA384 phase out ... AES128-GCM-SHA256 phase out ... AES128-SHA256 phase out ... AES256-SHA256 phase out For further Testexplanation please refere to https://www.internet.nl/site/www.covidcertificate.admin.ch/1249322/# Es versteht sich von selbst, dass bei so heiklen Vorhaben (politisch, applikatorisch & infrastrukturtechnisch, reputaionstechnisch) keine löchrigen Chiffrierroutinen verwendet werden sollten.	Die NCSC-NL taxiert in ihrem Tool www.internet.nl als gute Chiffrierroutinen: Good: ECDHE-ECDSA-AES256-GCM-SHA384 (TLS_AES_256_GCM_SHA384 in 1.3) [1.2] ECDHE-ECDSA-CHACHA20-POLY1305 (TLS_CHACHA20_POLY1305_SHA256 in 1.3) [1.2] ECDHE-ECDSA-AES128-GCM-SHA256 (TLS_AES_128_GCM_SHA256 in 1.3) [1.2] ECDHE-RSA-AES256-GCM-SHA384 (TLS_AES_256_GCM_SHA384 in 1.3) [1.2] ECDHE-RSA-CHACHA20-POLY1305 (TLS_CHACHA20_POLY1305_SHA256 in 1.3) [1.2] ECDHE-RSA-AES128-GCM-SHA256 (TLS_AES_128_GCM_SHA256 in 1.3) [1.2]		Analysis	
6	Hardcoded password	02.06.2021	Hardcoded Password	Hardcoded Password	the following code is vulnrable to Hardcoded password: https://github.com/admin-ch/CovidCertificate-Management-Service/blob/c1965356760315ce618b25c3db390d820bc503a2/src/main/java/ch/admin/bag/covidcertificate/CCManagementServiceApplication.java the password is "changeit"	https://github.com/admin-ch/CovidCertificate-Management-Service/blob/c1965356760315ce618b25c3db390d820bc503a2/src/main/java/ch/admin/bag/covidcertificate/CCManagementServiceApplication.java	Asaf Feigenbaum	Analysis	

					Hardcoded passwords may compromise system security in a way that cannot be easily remedied. It is never a good idea to hardcode a password. Not only does hardcoding a password allow all of the project's developers to view the password, it also makes fixing the problem extremely difficult. Once the code is in production, the password cannot be changed without patching the software. If the account protected by the password.				
7	Cypher Security	02.06.2021		WS supports weak ciphers	The acceptance environment supports weak ciphers, such as TLS_RSA_WITH_AES_128_CBC_SHA256 Since the prod env doesn't seem to be ready, I cannot confirm that this will be an issue there too. For more details see: https://www.ssllabs.com/ssltest/analyze.html?d=ws.covidcertificate-a.bag.admin.ch&hideResults=on		ozzi	Analysis	

8	Unbounded Query	02.06.2021		Unbounded Query when retrieving the revocation list	The implementation of findAllUvcis has an unbounded query. This is not an issue for 1000s of revocations, but considering that entire lots of certain vaccines could become known as ineffective, the number of revoked certificates may quickly increase by 10^5 certificates weekly. Devices calling this endpoint even in small numbers on a daily basis will potentially DoS the system, exceed any limits for response body sizes and provoke timeouts.
					
					@Repository
					public interface RevocationRepository extends JpaRepository<Revocation, UUID> {
					Revocation findByUvci(String uvci);
					
					@Query("SELECT r.uvci FROM Revocation r")
					List<String> findAllUvcis();
					}
							enzian	Analysis	

9	Use of signature is not thread-safe	02.06.2021		Use of signature is not thread-safe	It seems the use of Signature in the CryptoController is not thread-safe and parallel request could result in wrong signatures to be returned to the caller. The Signature is initialized as a Singleton bean in		Initdch	Fixed	

					CovidCertificate-Signing-Service/src/main/java/ch/admin/bag/covidcertificate/signature/config/HsmConfig.java
					
					Line 63 in bb7af92
					Signature signingSignature(KeyStoreEntryReader keyStoreEntryReader) {
					
					So the signature is shared between all the requests and this could lead to wrong requests or a SignatureException when the update message has been reset by another call. Signature (JDK 11)
					
					CovidCertificate-Signing-Service/src/main/java/ch/admin/bag/covidcertificate/signature/web/controller/CryptoController.java
					
					Lines 23 to 24 in bb7af92
					this.signingSignature.update(message);
					return this.signingSignature.sign();				

10	Embedded credentials	03.06.2021	Embedded passwords for the PostgreSQL DB	Embedded passwords for the PostgreSQL DB created in docker ./CovidCertificate-Management-Service/docker/docker-compose.yml environment: - POSTGRES_USER=cc-management - POSTGRES_PASSWORD=secret - POSTGRES_DB=cc-management	PostgreSQL in this specific docker configuration (version 13) also allows remote code execution of payloads Metasploit module used: (linux/postgres/postgres_payload) On some default Linux installations of PostgreSQL, the postgres service account may write to the /tmp directory, and may source UDF Shared Libraries from there as well, allowing execution of arbitrary code. This module compiles a Linux shared object file, uploads it to the target host via the UPDATE pg_largeobject method of binary injection, and creates a UDF (user defined function) from that shared object. Because the payload is run as the shared object's constructor, it does not need to conform to specific Postgres API versions.	Perhaps it should be part of each docker image to generate some random passwords so that these are unique to each instance.	Pavel Jirout	Analysis	

					http://www.leidecker.info/pgshell/Having_Fun_With_PostgreSQL.txt https://nvd.nist.gov/vuln/detail/CVE-2007-3280 Currently this impacts the docker postgresql image ./CovidCertificate-Management-Service/docker/docker-compose.yml services: db-cc-management: image: postgres				
11	Embedded passwords	03.06.2021	CovidCertificate-Signing-Service has some embedded passwords	CovidCertificate-Signing-Service has some embedded passwords which should be handled differently There are a few entries of embedded credentials Searching for a simple "secret" string on the git cloned sources reveals user@CF31:~/WORK/SWISS-HACKATON-COVID/CovidCertificate-Signing-Service$ grep -R secret . ./src/test/resources/application-test.properties:server.ssl.key-store-password=secret ./src/test/resources/application-test.properties:server.ssl.key-password=secret ./src/test/resources/application-test.properties:server.ssl.trust-store-password=secret ./src/test/resources/application-test.properties:app.signing-service.monitor.prometheus.password={noop}secret ./src/test/java/ch/admin/bag/covidcertificate/signature/web/controller/CryptoControllerIntegrationLocalTest.java: .keyStore(getFile(keystore), "secret") ./src/test/java/ch/admin/bag/covidcertificate/signature/web/controller/CryptoControllerIntegrationLocalTest.java: .trustStore(getFile(keystore), "secret"); ./src/test/java/ch/admin/bag/covidcertificate/signature/web/controller/CryptoControllerIntegrationTest.java: "app.signing-service.monitor.prometheus.password={noop}secret"})	Secrets are shown		Pavel Jirout	Analysis	

				./src/main/resources/application-local.properties:server.ssl.key-store-password=secret ./src/main/resources/application-local.properties:server.ssl.key-password=secret ./src/main/resources/application-local.properties:server.ssl.trust-store-password=secret ./src/main/resources/application-local.properties:app.signing-service.monitor.prometheus.password={noop}secret ./src/main/java/ch/admin/bag/covidcertificate/signature/service/KeyStoreEntryReader.java: private static final String MOCK_AND_TEST_PASS = "secret"; ./src/main/java/ch/admin/bag/covidcertificate/signature/config/HsmMockConfig.java: private static final String KEY_STORE_PASSWORD = "secret"; Interesting files containing plaintext creds are ./src/main/resources/application-local.properties ./src/test/resources/application-test.properties ./src/main/resources/application-local.properties					
12	Embedded passwords	03.06.2021	CovidCertificate-Management-Service has some embedded plaintext passwords	CovidCertificate-Management-Service has some embedded plaintext passwords namely the "secret" string when recursively searched through the git sources Affected files ./CovidCertificate-Management-Service/src/main/resources/application.yml ./CovidCertificate-Management-Service/src/main/resources/application-local.yml ./CovidCertificate-Management-Service/target/classes/application.yml ./CovidCertificate-Management-Service/target/classes/application-local.yml	plaintext secrets readable		Pavel Jirout	Analysis	
13	Mobile Apps: WebView JavaScript enabled	04.06.2021	Bei beiden mobile Apps Covid Check und Covid Cert wird im File ch/admin/bag/covidcertificate/co	Standardmässig ist JavaScript in WebView deaktiviert, falls nicht explizit benötigt sollte dies deaktiviert werden. OWASP beschreibt die Auswirkung einer Aktivierung wie folgt: "JavaScript can be injected into web applications via reflected, stored, or DOM-based Cross-Site Scripting		Falls möglich, JavaScript im WebView nicht aktivieren und die entsprechende Zeile löschen oder auf false setzen. Weitere Infos unter:	0x142	Analysis	

			mmon/html/HtmlFragment.java JavaScript für WebView aktiviert mittels: settings.setJavaScriptEnabled(true)	(XSS). Mobile apps are executed in a sandboxed environment and don't have this vulnerability when implemented natively. Nevertheless, WebViews may be part of a native app to allow web page viewing. Every app has its own WebView cache, which isn't shared with the native Browser or other apps. On Android, WebViews use the WebKit rendering engine to display web pages, but the pages are stripped down to minimal functions, for example, pages don't have address bars. If the WebView implementation is too lax and allows usage of JavaScript, JavaScript can be used to attack the app and gain access to its data." - https://github.com/OWASP/owasp-mstg/blob/1.1.3/Document/0x05h-Testing-Platform-Interaction.md#testing-javascript-execution-in-webviews-mstg-platform-5		https://github.com/OWASP/owasp-mstg/blob/1.1.3/Document/0x05h-Testing-Platform-Interaction.md#testing-javascript-execution-in-webviews-mstg-platform-5			
14	http api au lieu de https	07.06.2021	Le code github propose un site pour les api en http au lieu de https.	https://github.com/admin-ch/CovidCertificate-Apidoc/blob/main/api-doc.json	Cela pourrais poser un problème, car les informations envoyées passent en clair, et une option pour recouvrer un certificat, existe dans le code.	passer en https évitera l'interception des données		Fixed	
15	Timestamp issue	11.6.2021	Certificate issue timestamp not adapted to daylight saving time	Im Zertifikat steht "Zertifikat erstellt am 08.06.2021 um 21:08", SMS von vacme.ch kam um 21:23 (local time, also UTC+2h), ich holte es um 21:26. Das PDF hat intern einen Timestamp 20:08:06. Vermutlich läuft die Plattform mit einem festen UTC-Offset, d.h. hat noch Winterzeit.		Korrekter Timestamp oder korrekte Systemzeit	Andrea Schlapbach	Won't fix	Not reproducible
16	Generation API accepts nonvisible characters	12.6.2021	Generation API accepts nonvisible characters (unbreakable spaces, newlines , etc.) which relocate or hide information in the pdf.	The Generation and Revocation API for system integrators (integration with one-time-passwords) allows the usage of nonvisible characters for the following data fields: • familyName • givenName The API generates valid CovidCert-PDF-Documents, that have shifted names or no person identifying information at all. See some JSON input examples in the attachment. Chinese Characters and Emojis show a similar behaviour. They are not visible in the PDF, but displayed in the Verifier-App.	Such documents could invite people to fill in their names and the document could be accepted when the verifier only checks the paper version (and does not use the Verifier-App to check the information in the QR-Code). The existence of such CovidCert-PDF-Documents could reduce the trust in the CovidCerts and disrupt verification processes.	Define a list of permitted character and character chains (at least 2 visible characters should be mandatory) and filter the input accordingly.	Annett Laube	Analysis	
17	App non disponible si pays du profil utilisateur Google Play n'est pas en Suisse	17.6.2021	App non disponible si pays du profil utilisateur Google Play n'est pas en Suisse	Bonjour, Il s'agit ici d'un bug, plutôt qu'une faille de sécurité. Mon profil Google Play est paramétré sur "France" (profil de paiement). La recherche de l'app Covid Certificate dans Google Play à partir de mon	Application non disponible au téléchargement pour des profils utilisateurs Google Play défini en dehors de la Suisse (ou du moins si paramétré comme étant en France).	Rendre le téléchargement sur Google Play indépendant de la localisation géographique, que ce soit par l'adresse IP ou le paramétrage du profil GG Play.		Analysis	

				smartphone fait apparaitre que l'application n'est pas disponible dans ma région (alors que je suis connecté en WIFI sur une ligne fixe à Yverdon-les-Bains). La recherche sur Google Play depuis mon PC connecté à la même adresse IP, mais sans être loggé, fait bien apparaitre l'application à télécharger. Dès que je me log sur mon PC avec mon profil Google, l'application disparait de la liste et n'est plus disponible au téléchargement. Conclusion: la disponibilité de l'application dépend du sign in, et donc de la localisation paramétrée dans l'application Google Play, et non pas de l'adresse IP					
18	Access Token Logging in Production	17.6.2021	Im Management-Service wird das Loglevel für "ch.admin.bag.*" global auf DEBUG gesetzt	Im Management-Service wird das Loglevel für "ch.admin.bag.*" global auf DEBUG gesetzt. https://github.com/admin-ch/CovidCertificate-Management-Service/blob/6d7bbd85ab75b1ea603e2112c02758ce1769a260/src/main/resources/application.yml#L13 Das Loglevel wird nicht spezifisch pro Umgebung überschrieben und führt dadurch dazu, dass sensitive Daten (Access Tokens) auf die Console geloggt werden. https://github.com/admin-ch/CovidCertificate-Management-Service/blob/47cfe9bdf9b9eee9f14639bd703b4c2eaad8d1c7/src/main/java/ch/admin/bag/covidcertificate/web/controller/SecurityHelper.java#L22 https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html#data-to-exclude	Access Tokens können aus den Logs kopiert und verwendet werden, um sich als jemand anderes auszugeben.	Access Tokens sollten nur ohne Signatur geloggt werden oder das Loglevel entsprechend angepasst werden. Achtung beim Umstellen, die Nachvollziehbarkeit wer wann welches Zertifikat ausgestellt hat geht ohne Anpassung verloren. https://github.com/admin-ch/CovidCertificate-Management-Service/blob/6d7bbd85ab75b1ea603e2112c02758ce1769a260/src/main/java/ch/admin/bag/covidcertificate/web/controller/CovidCertificateGenerationController.java#L48	Jw Martin	Analysis	
19	iText Version Legacy	18.6.2021	EOL Software für PDF Generierung	Backend-seitig wird offenbar eine Version von iText zur PDF-Kreierung verwendet, welche Stand heute bereits End-of-Life ist. Die Informationen dazu sind in den Metadaten der erstellten PDF sichtbar.	EoL Software wird nicht mehr unterhalten und erhält Üblicherweise auch keine Patches mehr für Sicherheits-Lücken. Entsprechend entsteht das Risiko, dass in der PDF-Generierung nicht behobene Vulnerabilities bestehen. Da sämtlicher Input für die PDF-Generierung durch die Applikation selbst erzeugt wird, ist das entstehende Risiko klein. Es wärde lediglich dann zu einem Problem werden, wenn nicht validierter Benutzer-Input in das PDF Übernommen wird.	Auch wenn kein nicht validierter Input in das PDF Übernommen wird, empfiehlt es sich eine aktuelle Version von iText zu verwenden. Dies einerseits weil die Verwendung von EoL Software bei Go-Live immer vermieden werden sollte, andererseits auch um allfällig künftigen funktionalen Erweiterungen (welche eben diese Vulnerability triggern könnten) vorzubeugen.		Won't fix	Version 5 is the latest open source version. Security releases still happen for iText5.

20	Holder PII is logged	21.6.2021	DGC holder PII is stored in logs	PII of the COVID certificate holder is contained in log messages in the source code, for instance in the Cert Generation backend. This way, a central data collection of all holders is created, which is not intended and not allowed by the EDÖB.		All log messages should be reviewed to not contain any holder PII. Log messages purely intended for testing purposes should be marked accordingly and removed before production or publishing.	NTC	Fixed	
21	Wrong algorithm used for COVID certificate document signing	21.6.2021	A wrong COVID certificate document signing algorithm is used, which makes interoperability impossible.	The EU specification demands RSASSA-PSS 10 with SHA-256 (COSE PS256 11) and 2048-bit key length, whenever RSA keys are used. However, both the production and the mock implementation use different mechanisms.		Both implementations (production and mock) should use the SHA256withRSAandMGF1 (PSS padding) algorithm with 2048-bit RSA keys.	NTC	Fixed	
22	Android apps use hardcoded keys for COVID certificate validation	21.6.2021	Android apps use hardcoded keys for COVID certificate validation	The Android verifier and wallet apps have hardcoded keys for COVID certificate validation.	Having hardcoded keys, instead of fetching them from a backend service whenever an Internet connection is possible, leads to the impossibility of revoking signing keys without an update of the apps. In other words, whenever a single key is revoked, a new version of the app must be deployed to all devices.	It is recommended to implement a synchronization of the trusted keys at runtime whenever an Internet connection is available to the mobile device. The trusted keys should be stored securely within Android to prevent an attacker with access to the apps sandbox from altering the trusted keys list. Additionally, the apps can have a hardcoded public key of the Swiss root trust certificate to validate the keys within the trusted keys list. (This requires a proper public-key infrastructure by the issuer.)	NTC	Fixed	
23	iOS apps use hardcoded keys for COVID certificate validation	21.6.2021	The iOS verifier and wallet apps have hardcoded keys for COVID certificate validation.	The Android verifier and wallet apps have hardcoded keys for COVID certificate validation. For the validation of COVID certificates, the public key of the respective issuer is required on the mobile apps. Due to the requirement that the validation of COVID certificates must be possible offline, the respective key must be fetched from a backend service and stored securely on the mobile device. In the Swiss implementation of the verifier and wallet app, the CovidCertificateSDK is integrated. This SDK contains a hardcoded list of trusted public keys for the COVID certificate validation.	Having hardcoded keys, instead of fetching them from a backend service whenever an Internet connection is possible, leads to the impossibility of revoking signing keys without an update of the apps. In other words, whenever a single key is revoked, a new version of the app must be deployed to all devices. A centrally revoked signing key, e.g., due to compromise or lifetime expiry, would not be recognized by the mobile apps. Both apps would still recognize COVID certificates issued by the revoked signing key as valid. An update of the mobile apps would be required to update the hardcoded keys list.	It is recommended to implement a synchronization of the trusted keys at runtime, whenever an Internet connection is available to the mobile device. The trusted keys should be stored within the iOS Keychain to prevent an attacker with access to the apps sandbox, from altering the trusted keys list. Additionally the apps can have a hardcoded public key of the Swiss root trust certificate, to validate the keys within the trusted keys list. (This requires a proper public key infrastructure by the issuer.)	NTC	Fixed	
24	OTP validation bypass	21.6.2021	The ImmunityRequest API has a hardcoded test phone number and test OTP code to bypass validation.	The security design of the recovery COVID certificate process demands that a user must enter a mobile phone number to retrieve an OTP code. This OTP code should then be verified by the backend before accepting the submitted immunity request form. However, it was found that a hardcoded phone number cs.testPhone and a hardcoded OTP code cs.testSMSCode can be used to bypass this process.	An attacker who is in the possession of the cs.testPhone and cs.testSMSCode values can circumvent the validation process. This can lead to mass immunity request form requests. Depending on the number of NO_MATCH or PARTIAL_MATCH entries, the cantons might not be able to fulfill their tasks, to differentiate legitimate from non-legitimate requests. The entire issuing process may get throttled by an unacceptable delay.	It is recommended to remove the code responsible for the OTP validation bypass. Alternatively, an additional check for the actual deployment environment can be implemented in the by-pass logic to keep the capabilities for TEST, DEV, and ABN environments.	NTC	Fixed	
25	CSRF on CSV import	21.6.2021	The ImmunityRequest API is	The immunity request form is not protected against Cross-Site-Request-Forgery (CSRF) attacks. A CSRF attack invokes the	The web server and the web application is not able to distinguish a valid request from the request forged through CSRF. Valid	It is recommended to implement an anti-CSRF token. Such a token must be linked to the user session at the backend. The	NTC	Fixed	

			vulnerable to a Cross-Site-Request-Forgery (CSRF) attack on the CSV import function.	victim's browser to silently send requests to the immunity request form whenever the victim opens an attacker-controlled website. The only precondition is that the victim is logged in at the immunity request form in the role of a canton user.	credentials, e.g., the pams-abn cookie, are automatically added by the browser when the request is sent. One possible scenario is that the victim visits the attacker's website, which will then automatically and silently send the GET or POST request to the immunity web application. The form for the GET or POST request is automatically submitted and typically invisible to the victim. The HTTP request will execute the attacker's action.	token must be robust against guessing and brute-forcing. The token must be validated on the server side for every HTTP POST request. The token must not be sent as a cookie, but as a custom HTTP header, e.g., X-Csrf-Token. If present, it is recommended to use standard libraries accessible to the server environment.			
26	Private key is used to verify JWT signature	21.6.2021	The private key is used to verify the signature of JSON Web Tokens. Instead, the public key should be used. As a consequence, the private JWT signing key is stored in the API Gateway Service instance.	For authentication on the Machine-to-Machine interface, alongside mTLS, an OTP is used in form of a signed JSON Web Token (JWT). This token is signed by the cc-management-service with an RSA private key as defined in ${"cc-management-service.jwt.privateKey"} in the YAML configuration file of the cc-management-service. When validating this JWT and verifying its signature in the cc-api-gateway-service, the corresponding public key should be used. However, we found that the private key, as stored in ${"cc-api-gateway-service.jwt.privateKey"} , was used to verify the signature.	As mentioned above, this design comes with a number of risks: • The design is unusual and may lead to unexpected consequences. The private key could be compromised by an attacker with access to the verification instance, and be then used to sign arbitrary JSON Web tokens. In other words, the verification instance can be abused for signing arbitrary tokens. These tokens can be used to authenticate on the Machine-to-Machine interface. • The private signing key should not be stored in the API Gateway Service instance, as only the corresponding public key is necessary to validate the JWT signature. • The private verification key would have to be shared among various instances in case the workload is balanced. In this case, the private key becomes a shared key and needs more protection.	It is recommended to only use the public key to verify the signature of any JWT token. The private key is only used for signing the JWTs, and should be kept in a secure location. When mitigating this risk, it is recommended to remove the private key from the current storage location cc-api-gateway-service.jwt.privateKey in resources/application-ENV.yml , and replace it solely by its corresponding public key.	NTC	Fixed	
27	Missing link of canton user and UVCI	21.6.2021	The function Valid returns a wrong error message if an invalid PatientZipCode is provided.	The immunity request form allows canton users to issue COVID certificates by approving and re-uploading CSV files. A malicious canton user can, approve arbitrary COVID certificate requests. In order to be able to revoke (blacklist) such COVID certificates retrospectively a link between the canton user (issuer) and the UVCI of the certificates must be logged. (As implemented in the Covid Certificate Management Service).	As the ImmunityRequest-API is treated as an Primary System, the Covide Certificate Management Service will not create this link between the actual canton user (issuer) and the generated UVCI. However, the following must be taken into account, the ImmunityRequest-API can create an OTP for the interface to the Covide Certificate Management Service itself, since access to the private key is granted. (see section Interface Certificate Management Service VACCINECER-319211342-090621-1843-834.pdf). In case that a malicious canton user is detected, the current implementation may result in a blacklisting of all COVID certificates issued by the canton.	It is recommended to create a link between the canton user (issuer) and the generated UVCI, by logging this information in Splunk. This recommendation applies to all Primary Systems, which are connected to the Covide Certificate Management Service in the same way.	NTC	Fixed	
28	Potentially insecure TLS setting	21.6.2021	The ImmunityRequest API allows for insecure TLS connections.	The ImmunityRequest API allows for insecure TLS connections by setting the parameter InsecureSkipVerify to true. This can be seen in an extract from lib/requestlib/genb_send.go.	As the documentation states, having the InsecureSkipVerify parameter set to true allows an attacker to perform a Machine-in-the-Middle attack.	It is recommended to always set the parameter InsecureSkipVerify to false.	NTC	Fixed	

29	JWT audience validation deviates from RFC7519	21.6.2021	The validation of the JWT audience is not compliant with RFC7519. An empty audience will be accepted.	The validation of the OAuth2.0 JWT contains a check for the audience parameter. The actual implementation can lead to an empty audience being validated.	It is not assumed that this issue can be exploited in practice. Nevertheless, it is recommended to follow the RFC considerations.	According to the RFC considerations, the implementation can be as follows:
						
						1 public OAuth2TokenValidatorResult validate(Jwt jwt) {
						2 if(jwt.getAudience () == null) {
							NTC	Fixed	

						3 return OAuth2TokenValidatorResult.success();
						4 }
						5 if (jwt.getAudience ().contains(audience)) {
						6 return OAuth2TokenValidatorResult.success();
						7 } else {
						8 log.warn(error.getDescription ());
						9 return OAuth2TokenValidatorResult.failure(error);
						10 }
						11 }			

30	M2M test TLS certificate valid for production	21.6.2021	The WAF WSG configuration for the M2M access uses the same root certificate for acceptance and production.	The Webservice Gateway (WSG) configuration for the M2M interface on the Web Application Firewall (WAF) uses the same root certificate Swiss Government Regular CA 01 for validation of both the acceptance (ws.covidcertificate-a.bag.admin.ch) and the production (ws.covidcertificate.bag.admin.ch) access points.	This allows anyone with access to a (much less controlled) test certificate to access at least the general M2M production endpoints.	For test access, a separate root certificate should be used to clearly separate the use-cases.	NTC	Fixed	
31	Downloaded revocation list cannot be verified	21.6.2021	The list of revoked UVCIs, which can be downloaded from the CovidCertificate management service, is not signed.	The list of revoked UVCIs22, as provided by the CovidCertificate management service endpoint/api/v1/revocation-list, contains no application-layer integrity protection.	If no application layer integrity protection mechanism such as JWS or another cryptographically signed data container is in place, a component that is positioned between the consumer and the creator of the revocation list could modify the content. Especially the external content delivery network (CDN) could potentially add or remove UVCIs.	It should be considered to have the revocation list be cryptographically signed by the creating component. This way, the required trust in external systems can be reduced. This should be considered as a defense-in-depth mechanism additional to the authenticated communication via TLS.	NTC	Fixed	