Virgil PureKit

Enable post-compromise protection for stored data with Virgil Security framework and MariaDB database.

Dmitry Dain
Founder & CTO at Virgil Security
Data protection is **NOT** optional anymore

- Data breaches: 7.9 billion records exposed in 2019
- Acts and laws compliance: GDPR, HIPAA, PCI DSS, CCPA
- High fines: e.g. British Airways — around $230 million
## Available solutions for secure data storage

<table>
<thead>
<tr>
<th>Features</th>
<th>TDE</th>
<th>RDS Encryption</th>
<th>PureKit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key distribution</td>
<td>Per database</td>
<td>Per database</td>
<td>Per user/client</td>
</tr>
<tr>
<td>Key rotation</td>
<td>Database re-encryption required</td>
<td>Database re-encryption required</td>
<td>In-place</td>
</tr>
<tr>
<td>Post-compromise security</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL injection secure</td>
<td>No (data is still selected as plaintext)</td>
<td>No (data is still selected as plaintext)</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### What’s post-compromise security?

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seamless key rotation</td>
<td>- Keys can be rotated while app is running</td>
</tr>
<tr>
<td></td>
<td>- Proactively or in case of data breach</td>
</tr>
<tr>
<td>Data protected even if database is compromised</td>
<td>- Instant stolen data invalidation</td>
</tr>
<tr>
<td>Zero-knowledge proof</td>
<td>- Crypto service proves all operations were performed using its private key</td>
</tr>
<tr>
<td>Online and offline attacks are not possible</td>
<td>- Strict rate limiting per-user</td>
</tr>
<tr>
<td></td>
<td>- Both App &amp; Crypto service private keys required to guess password</td>
</tr>
</tbody>
</table>
How about AWS KMS?

Key Management System
Key Management System (KMS)

AWS KMS saves the CMK’s older cryptographic material in perpetuity

Single master symmetric key for all crypto operations
# Key Management System (KMS)

<table>
<thead>
<tr>
<th>Features</th>
<th>AWS</th>
<th>Virgil Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDPR/HIPAA compliance</td>
<td>✗</td>
<td>+</td>
</tr>
<tr>
<td>Single point of failure</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>User controls data access</td>
<td>✗</td>
<td>+</td>
</tr>
<tr>
<td>Post-compromise secure</td>
<td>✗</td>
<td>+</td>
</tr>
<tr>
<td>Insider-secure</td>
<td>✗</td>
<td>+</td>
</tr>
<tr>
<td>Price</td>
<td>1 CloudHSM is over $1K/month</td>
<td>Cloud KMS Pricing</td>
</tr>
</tbody>
</table>
Understanding Virgil PureKit and ZKP
## PureKit

Open-source security framework for developers to enable post-compromise protection for stored data

<table>
<thead>
<tr>
<th>Functions:</th>
<th>Benefits:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Per-user data and files encryption</td>
<td>User controls data access</td>
</tr>
<tr>
<td>Role-based data encryption</td>
<td>Post-compromise security</td>
</tr>
<tr>
<td>Secure data sharing</td>
<td>GDPR/HIPAA compliance</td>
</tr>
<tr>
<td>Key rotation</td>
<td>Works with any database</td>
</tr>
<tr>
<td>Password-hardened encryption</td>
<td>Low overhead, instant “re-encryption”</td>
</tr>
<tr>
<td>Data access recovery</td>
<td></td>
</tr>
</tbody>
</table>
How PureKit works

All user’s keys are protected with a unique user’s PURE Record

Application Users

Database Admins

Your Server

Database

PureKit

Virgil Cloud
Cryptographic Keys Service

Services of statistics, analytics, etc.

Third Party Services
How PureKit Works

Phase #1. The App Backend asks PHE Service for an Enrollment

App Backend:

1. Sends empty request to the PHE service.

PHE Service:

1. Generates 32-byte random salt.

2. Hashes salt with two different domains into two curve points $HS_0$ and $HS_1$.

3. Performs scalar multiplication of $HS_0$ and $HS_1$ by its Private Key ($Y$) to get points $C_0$ and $C_1$.

4. Calculates Zero Knowledge Proof which proves that $C_0$ and $C_1$ were indeed calculated using app server’s Private Key ($Y$).

5. PHE Service replies with the following data:
   a. 32-byte random salt
   b. Points $C_0$ and $C_1$
   c. ZKP
### Pure Record

A unique data that is associated with a specific user’s password, 1 password = 1 record

**Pure Record composition:**

<table>
<thead>
<tr>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>User’s Pure Record version</td>
</tr>
<tr>
<td>App &amp; PHE service random salts</td>
</tr>
<tr>
<td>Two values obtained during the execution of the PHE protocol</td>
</tr>
<tr>
<td>User’s PHE Encryption key</td>
</tr>
</tbody>
</table>
PURE Record Enrollment

Create unique Record for each user on PHE service

PUBLIC KEY

PRIVATE KEY (x)

BACKEND

Enroll?

PRIVATE KEY (y)

PHE SERVICE

sNonce, C0, C1, Proof
C0, C1 – points
C0 = y · (sNonce, "0")
C1 = y · (sNonce, "1")

HCO = x · (cNonce, password, "0")
HC1 = x · (cNonce, password, "1")
M – random, MC = x · M

T0 = C0 + HCO
T1 = C1 + HC1 + MC

DATABASE:
T0, T1, sNonce, cNonce
PURE Record Verification

Verifies user’s Record in the database during login step

**BACKEND**
- PRIVATE KEY (x)

**PHE SERVICE**
- PRIVATE KEY (y)

\[ C_0, sNonce \]
\[ C_1, \text{Proof} \]

\[ H_{C0} = x \cdot (cN nonce, \text{password}, "0") \]
\[ C_0 = T_0 - H_{C0} \]

\[ H_{C1} = x \cdot (cN nonce, \text{password}, "1") \]
\[ M_C = T_1 - C_1 - H_{C1} \]

\[ C_0 == y \cdot (sN nonce, "0") ? \]
How PureKit + MariaDB plugin works

Application → Encryption Keys → PureKit with MariaDB plugin → Database

Post-compromise security enabled

Data is protected

MariaDB Database

Cloud Storage Provider

Integration with 3rd Parties

Your App Service Provider

Integration with 3rd Parties

COMPLIANCE

INVENTORY

ANALYTICS

BILLING
Pre-release version includes

**Password protection**
- Password hashing is replaced in a way that it's impossible to run offline and online attacks

**Per-user data encryption**
- Each user owns personal data encryption key

Getting started with PureKit

Ask anything about MariaDB plugin based on PureKit
support@VirgilSecurity.com

PureKit documentation
Developer.VirgilSecurity.com