Table 1: An overview of four different user profiles based on seven different security focus areas is presented. Performance of each profile in each focus area is evaluated with the number from 1 - 5. A red arrow (▼) indicates that users are performing below average, while a green arrow (▲) indicates above average results. Profile 1 with 65 users, for example, is poor (below average) in all security areas while Profile 2 with 43 users is performing above the average in all areas.

Security areas / **User profiles**	**[PM]** Password management	**[EU]** Email use	**[IU]** Internet use	**[SMU]** Social media use	**[MDU]** Mobile devices use	**[IH]** Information handling	**[IR]** Incident reporting
Profile 1 (N = 65 users)	▼ 3.66	▼ 3.63	▼ 3.34	▼ 3.38	▼ 3.75	▼ 3.75	▼ 3.38
Profile 2 (N = 43 users)	▲ 4.23	▲ 4.43	▲ 4.36	▲ 4.37	▲ 4.85	▲ 4.72	▲ 4.35
Profile 3 (N = 35 users)	▲ 4.11	▼ 3.73	▲ 3.93	▲ 4.08	▲ 4.43	▲ 4.62	▲ 3.99
Profile 4 (N = 22 users)	▲ 3.99	▲ 4.43	▼ 3.79	▲ 3.94	▲ 4.44	▼ 4.06	▲ 4.07
Average	3.95	3.97	3.79	3.86	4.27	4.23	3.85

Table 2: A list of 15 security functionalities. The meaning of colors in the table below: **Green** (🟩) – security functionality comprehensively addresses the security area. **Yellow** (🟨) – security functionality partially addresses the security area. **White** (⬜) – security functionality does not address the security area. In practice, for example, no security area can be covered by "ID 1: Verify that the chosen …".

		PM	**EU**	**IU**	**SMU**	**MDU**	**IH**	**IR**
	Number of users that are **below average by focus areas**:	65	100	87	65	65	87	65
	Security functionality							
1	Verify that the chosen access control solution is flexible enough to meet the application's needs.							
2	Verify the application uses a single and well-vetted access control mechanism for accessing protected data and resources. All requests must pass through this single mechanism to avoid copy and paste or insecure alternative paths.							
3	Verify impersonation resistance against phishing, such as the use of multi-factor authentication, cryptographic devices with intent (such as connected keys with a push to authenticate), or at higher AAL levels, client-side certificates.							
4	Verify replay resistance through the mandated use of One-time Passwords (OTP) devices, cryptographic authenticators, or lookup codes.							
5	Verify intent to authenticate by requiring the entry of an OTP token or user-initiated action such as a button press on a FIDO hardware key.							
6	Verify that symmetric keys used to verify submitted OTPs are highly protected, such as by using a hardware security module or secure operating system based key storage.							
7	Verify that if a time-based multi-factor OTP token is re-used during the validity period, it is logged and rejected with secure notifications being sent to the holder of the device.							
8	Verify physical single-factor OTP generator can be revoked in case of theft or other loss. Ensure that revocation is immediately effective across logged in sessions, regardless of location.							
9	Verify that the application gives the option to terminate all other active sessions after a successful password change (including change via password reset/recovery), and that this is effective across the application, federated login (if present), and any relying parties.							
10	Verify that users are able to view and (having re-entered login credentials) log out of any or all currently active sessions and devices.							
11	Verify the application allows users to revoke OAuth tokens that form trust relationships with linked applications.							
12	Verify the application has additional authorization (such as step up or adaptive authentication) for lower value systems, and / or segregation of duties for high value applications to enforce anti-fraud controls as per the risk of application and past fraud.							
13	Verify that the application logs security relevant events including successful and failed authentication events, access control failures, deserialization failures and input validation failures.							
14	Verify that exception handling (or a functional equivalent) is used across the codebase to account for expected and unexpected error conditions.							
15	Verify that a "last resort" error handler is defined which will catch all unhandled exceptions.							