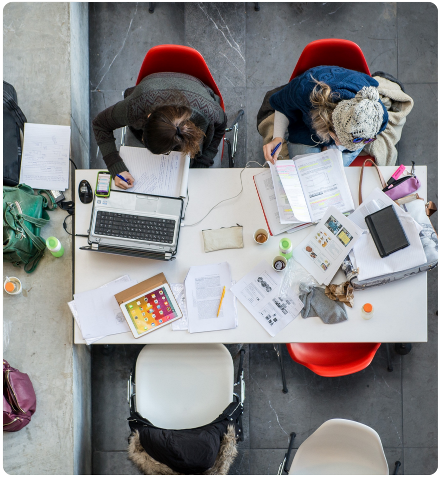
Programmable authentication workflows

Stefan Wolfsheimer, Claudio Cacciari

SURF

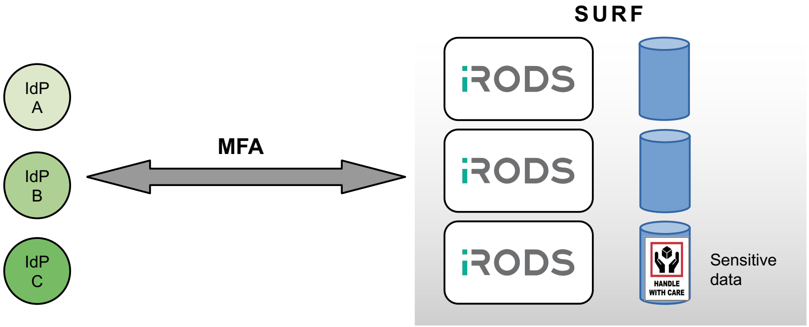
SURF is an ICT cooperative for education and research

At the SURF cooperative, education and research work together to make full use of the opportunities offered by digitalisation, with the aim: making education and research better and more flexible.



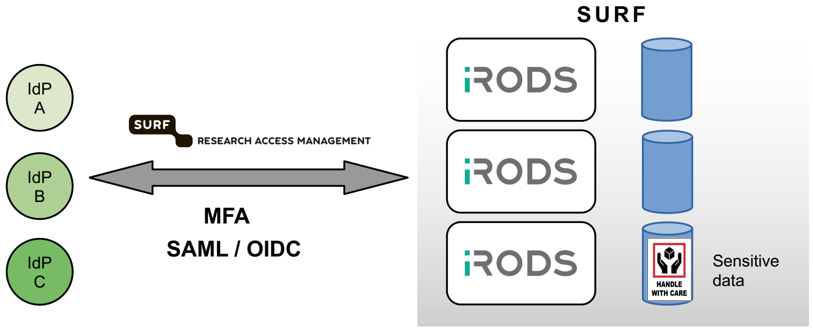
SURF Use case

- SURF offers iRODS as a hosted service.
- Increasing demand for multi-factor authentication (MFA) in front of the service,
- Option: Linked to institute's Identity Provider (IdP).



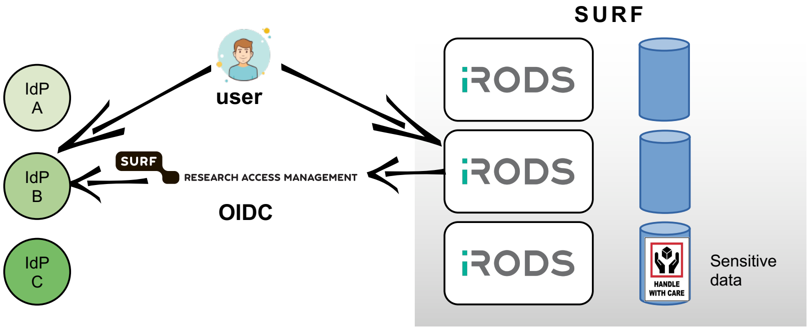
SURF Use case

- SRAM (SURF Research Access Management): Bridge between Identity and Service providers
- Protocols: OIDC, SAML
- MFA support



SURF Use case

- Integration: iRODS services with IdP via SRAM



iRODS authentication methods

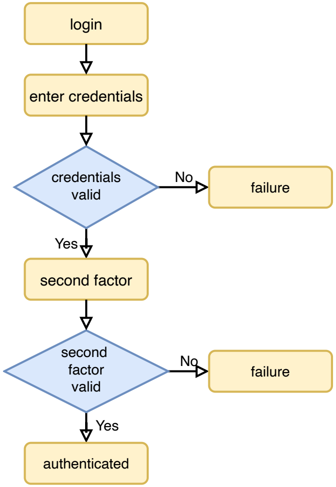
- Standard: native authentication (login and password)
- Plugins:
  - PAM
  - Kerberos
  - OIDC (?)
  - GSI (?)

Why yet another authentication plugin?

- Existing plugins not flexible enough
- No clear separation between iRODS and plugin code.

PAM: pluggable authentication module

- System admins implement flows by combining PAM modules
- Application developers implement PAM-enabled applications
- Module developers implement reusable blocks
- iRODS is PAM-enabled, but only standard user interaction supported



Example workflow: username password

- Examples based on `pam_python.so` module

```
#/etc/pam.d/irods
auth required pam_python.so /etc/pam.d/simple.py
# /etc/pam.d/simple.py
USERS_DB={
    'ayub': 'pw',
    'mara': 'ACtoRPHI',
    # ...
}

def pam_sm_authenticate(pamh, flags, argv):
    msg = pamh.conversation(pamh.Message(pamh.PAM_PROMPT_ECHO_ON, "login:"))
    pwd_msg = pamh.conversation(pamh.Message(pamh.PAM_PROMPT_ECHO_OFF, "password:"))

    if msg.resp in USERS_DB and pwd_msg.resp == USERS_DB[login]:
        return pamh.PAM_SUCCESS
    else:
        pamh.conversation(pamh.Message(pamh.PAM_TEXT_INFO,
                                       "unknown login {}".format(login)))
        return pamh.PAM_AUTH_ERR
```

Example workflow: second factor

```
#/etc/pam.d/irods
auth required pam_python.so /etc/pam.d/simple.py
auth required pam_python.so /etc/pam.d/2fa.py
```

```
#/etc/pam.d/2fa.py
def pam_sm_authenticate(pamh, flags, argv):
    # just prompt, don't save the pin locally
    payload = {"prompt": "enter pin:"}
    msg = pamh.conversation(pamh.Message(pamh.PAM_PROMPT_ECHO_ON,
                                         json.dumps(payload)))
    if msg.resp == "1234":
        # save token on client, no prompt
        token = str(uuid.uuid4())
        patch = {"patch": [{"op": "add",
                            "path": "/token",
                            "value": token}]}
        msg = pamh.conversation(pamh.Message(pamh.PAM_TEXT_INFO,
                                             json.dumps(patch)))
        return pamh.PAM_SUCCESS
    else:
        return pamh.PAM_AUTH_ERR
```

Example workflow: token exchange

```
def pam_sm_authenticate(pamh, flags, argv):
    # save token on client, no prompt
    token = str(uuid.uuid4())
    patch = {"patch": [{"op": "add",
                        "path": "/token",
                        "value": token}]}
    msg = pamh.conversation(pamh.Message(pamh.PAM_TEXT_INFO,
                                         json.dumps(patch)))
    return pamh.PAM_SUCCESS


def pam_sm_authenticate(pamh, flags, argv):
    # get token
         payload = {"retrieve": "/token"}
    msg = pamh.conversation(pamh.Message(pamh.PAM_PROMPT_ECHO_ON, json.dumps(payload)))
    if validate_toke(msg.resp):
        return pamh.PAM_SUCCESS
    else:
        return pamh.PAM_AUTH_ERR
```

Summary and outlook

Summary

- `pam_interactive`: programmable authentication schemes
- From simple username+password to complex flows
- Compatible with iRODS 4.3.0

Outlook and todos

- Test, review and validate in pre-production enviornments
- Extend client library support (python, Java)
- Explore use case / auth workflows

Thank you

- `https://github.com/stefan-wolfsheimer/irods_auth_pam_interactive.git`