Développement d'un plugin Powershell

Remarques importantes

- **Esia décline toute responsabilité** concernant l'utilisation de plugins non écrits par notre société. Certaines commandes powershell peuvent modifier ou détruire votre noeud Windows.
- **Esia peut refuser tout support** concernant un plugin qui n'a pas été créé par notre société.
- Avant de vous attaquer à la création de plugins powershell, poser toujours la question à Esia (Peut être qu'Esia possède déjà ce plugin ou souhaite l'ajouter prochainement).

Prérequis

- Posséder un accès ssh vers son Esia en tant qu'utilisateur root.
- Connaître un des langages de scripts compatibles Esia/Debian (php, perl, ...).

Ecriture de plugin Powershell

Ce tutoriel est un complément de l'article suivant (plus général sur l'écriture de plugin Esia : Comment créer un plugin (Uniquement pour ESIA Infinity)

Vous pouvez aussi créer directement un exécutable local à votre Windows, exécutable à distance via les plugins GESA_PWSH_REMOTE et CHECK_PWSH_REMOTE. Voir ce tuto : Plugin Powershell remote

Un démon Esia s'occupe de recevoir et d'envoyer toutes les demandes de connexions powershell. On peut lui envoyer des requêtes CURL. Mais uniquement en local (il écoute sur le 127.0.0.1 uniquement).

```
URL : http://127.0.0.1:2082
METHOD : POST
PARAMs :
key        : <secret partagé>  # entre le démon et les plugins. Différent
pour chaque Esia. Voir le champ 'key' du fichier INI suivant :
/etc/esia/rcm.ini
action    : run
proto     : ps1
hostaddr : <host_addr>
user : <host_user>
authentication : <host_userpass>
command : <La, les commandes Powershell a exécuter>
```

Exemples de commandes Powershell à exécuter :

```
Get-Counter '\Processor(_Total)\% Processor Time
```

ou bien

```
$hash = @{};
$hash.task=Get-ScheduledTask | select -Property $selectTask;
$hash.info=Get-ScheduledTask | Get-ScheduledTaskInfo | select -Property
$selectInfo;
$hash | ConvertTo-Json
```

L'avantage de cette dernière commande et d'exécuter un ensemble de commandes et de récupérer ensuite un json. Il est bien plus rapide/performant/moins gourmant en ressources d'envoyer une seule requête POST que plusieurs. En plus de l'exécution des commandes reçues, Windows prends beaucoup de temps à traiter la requête reçue ou bien générer la réponse à renvoyer.

Ci-dessous un exemple de script plugin Esia exécutant Powershell :

```
#!/usr/bin/php
<?php
include_once('/usr/local/esia/plugins/lib/HttpClient.class.php');

$DEFAULT_CONF_FILE = '/etc/esia/rcm.ini';
$DEFAULT_SRV_URL = 'http://127.0.0.1:2082';
$DEFAULT_TYPE = 'ps1';
$DEFAULT_METHOD = 'POST';

$STR_STATUS=array("OK","WARNING","CRITICAL","UNKNOWN");
$INT_STATUS=array_flip($STR_STATUS);
$PRIORITIES_STATES=array("CRITICAL","WARNING","UNKNOWN","OK");

$SHORT_OPTS="u:H:U:P:A:a:t:c:Dh";
$LONG_OPTS=array("url:","host:","user:","password:","authfile:",'action:','t
ype:','command:','debug','help');

$options = getopt($SHORT_OPTS, $LONG_OPTS);

$conf = parse_ini_file($DEFAULT_CONF_FILE,true);
$o_key = $conf['LISTEN']['key'];

if( isParam($options,'h','help') )
{
    print_help();
    exit($INT_STATUS['UNKNOWN']);
}

$o_method = $DEFAULT_METHOD;
$o_type = $DEFAULT_TYPE;
$o_debug = isParam($options,'D','debug');
$o_authfile_path = getParam($options,'A','authfile');

if( isset($o_authfile_path) )
{
    if( !file_exists($o_authfile_path) )
```

```
    {
        exit_state('UNKNOWN',"Unable to found -A, --authfile
$o_authfile_path");
    }
    $auth_file_data = file_get_contents($o_authfile_path);
    if( empty($auth_file_data))
    {
        exit_state('UNKNOWN',"Unable to open -A, --authfile
$o_authfile_path");
    }
    $auth_file_data = json_decode($auth_file_data,TRUE);
    if( !is_array($auth_file_data))
    {
        exit_state('UNKNOWN',"Unable to parse -A, --authfile
$o_authfile_path");
    }
}

$o_host = getParam($options,'H','host',@$auth_file_data['host']);
$o_user = getParam($options,'U','user',@$auth_file_data['user']);
$o_pass = getParam($options,'P','password',@$auth_file_data['password']);

if( empty($o_host) )
{
    exit_state('UNKNOWN',"Missing required option -H, --host or authfile
parameter host");
}
if( empty($o_user) )
{
    exit_state('UNKNOWN',"Missing required option -U, --user or authfile
parameter user");
}
if( empty($o_pass) )
{
    exit_state('UNKNOWN',"Missing required option -P, --password or authfile
paramter password");
}

$o_url = getParam($options,'u','url',$DEFAULT_SRV_URL);
$o_action = getParam($options,'a','action');

if( empty($o_url) )
{
    exit_state('UNKNOWN',"Missing required option -u, --url");
}
if( empty($o_action) )
{
    $o_action = 'run';
}

if( $o_debug )
```

```
{
    $params['debug'] = true;
    echo "DEBUG :\n";
    echo "URL : $o_method:$o_url\n";
    print_r($params);
    echo "END\n";
}



$params['key'] = $o_key;
$params['action'] = $o_action;
$params['proto'] = $o_type;
$params['hostaddr'] = $o_host;
$params['user'] = $o_user;
$params['authentication'] = $o_pass;
$params['command'] = 'Get-Service';
//$params['command'] = 'Get-EventLogs -LogName System -Newest 10| Select-
Object -Property Message';
//$params['command'] = 'Get-Process | Select-Object -Property Name';
//$params['command'] = 'Get-Process';
//$params['command'] = "Get-Counter '\Processor(_Total)\% Processor Time'";



$client = new HttpClient($o_url);
$client->open();
$client->setOptions(CURLOPT_CONNECTTIMEOUT,20);
$client->setOptions(CURLOPT_TIMEOUT,20);

$data = $client->post('',$params);
$code = $client->getHttpCode();
print_r(array($code,json_decode($data)));
$client->close();



echo "Ce plugin retournera toujours OK (0).\n";
exit($INT_STATUS['OK']);

function isParam($options,$name1,$name2)
{
    return isset($options[$name1]) || isset($options[$name2]);
}

function getParam($options,$name1,$name2,$default=null){
        if( isset( $options[$name1] ) ) return $options[$name1];
        if( isset( $options[$name2] ) ) return $options[$name2];
        return $default;
}

function exit_state($state,$msg=null,$usage=false){
        $STR_STATUS=$GLOBALS["STR_STATUS"];
        $INT_STATUS=$GLOBALS["INT_STATUS"];
```

```
        $ret=$state;
        if( is_string($state) ) $ret=$INT_STATUS[strtoupper($state)];
        else $state=$STR_STATUS[$ret];

        if( $msg != null )
        {
          $msg=trim($msg);
          echo "$state : $msg\n";
        }
        if( $usage ) print_usage(true);
        exit($ret);
}

function print_help()
{
        echo "
usage : check_pwsh.php -H <host> -U <user> -P <password -T <type> -w
<treshold> -c <treshold> [OPTIONS]
        check_pwsh.php -H <host> [-U <user>] -A <authfile> -T <type> -w
<treshold> -c <treshold> [OPTIONS]

        This function print_help() isn't important but can help users to
understand plugin.

    --h, --help                           print this help
    --D, --debug                          Use to enable debug mode
    --u, --uri=<url>                      Powershell daemon url
(http://127.0.0.1:2082 by default)
    --H, --host=<host>                    Host IP ou FQDN
    --U, --user=<login>                   Login or username
    --P, --password=<password/pathfile>   password or path for pubkey,
sharedkey or special authentication options
    --A, --authfile=<filepath>            authentication file filepath form
set default user, password parameters
    --w, --warning=<treshold>             warning treshold
    --c, --critical=<treshold>            critical treshold
";
}
?>
```