Creating a Honey Pot

By Bruce Thornton
10/12-13/2025

Professional Project Summary

Project Title: MX Linux Honeypot Integration with OpenCanary and (Eventual Configuration with Elastic Stack)
Role: Security Engineer / SOC Lab Developer
Duration: October 12-13, 2025
Overview:
Designed, built, and automated a production-grade honeypot environment using OpenCanary on MX Linux as part of a home SOC lab. The system was engineered for stealth, persistence, and seamless integration with Elastic Stack (via Kali Purple) for centralized monitoring and analysis.

Key Achievements:

- Deployed and configured OpenCanary v0.9.6 in a secure Python virtual environment using SysVinit for startup persistence on MX Linux (non-systemd).

- Implemented service hardening through least-privilege execution and Linux capabilities (`cap_net_bind_service`), enabling safe binding to privileged ports (21, 22, 80).

- Automated full startup lifecycle via custom SysV service scripts with log rotation, PID management, and error handling for `/var/run` tmpfs environments.

- Verified and tuned honeypot listeners (FTP, SSH, HTTP) for realistic attack surface simulation while preventing conflicts with legitimate daemons.

- Developed a lightweight rebuild checklist for rapid redeployment, including package installation, venv provisioning, init script registration, and UFW firewall configuration.

- Prepared system for syslog forwarding and integration with Elastic/Kibana dashboards through Kali Purple for live threat telemetry and alerting.

- Built the foundation for VLAN and pfSense segmentation to isolate honeypot traffic and safely simulate external attacks.

Outcome:
Achieved a stable, resource-efficient honeypot deployment that auto-starts at boot, provides actionable telemetry to Elastic Stack, and forms the core of a scalable, modular cybersecurity lab.

Selecting the Environment

This Honey Pot has been created using:

Lenovo ThinkCentre M93P Tiny Desktop, Intel Core i5-4570T, 8GB RAM, 256GB SSD, Windows 10 Pro 64-bit (Renewed)

Utilizing a USB drive, I have flashed MX Linux onto the USB drive.

🛠️ MX Linux + OpenCanary (Single-Page Rebuild Guide)
1️⃣ Install MX Linux (clean base)

Flash ISO to USB, reinstall MX Linux (no need for systemd).

After first boot:

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y git python3-venv python3-pip libcap2-bin ufw
```

2️⃣ Create virtual env and install OpenCanary

```
sudo mkdir -p /opt/opencanary
cd /opt/opencanary
sudo python3 -m venv venv
sudo chown -R $USER:$USER venv
source venv/bin/activate
pip install --upgrade pip setuptools==68.0.0
pip install opencanary==0.9.6
deactivate
```

3️⃣ Initialize config

```
sudo mkdir -p /etc/opencanaryd
sudo /opt/opencanary/venv/bin/python3 -m opencanaryd --copyconfig
sudo chown -R opencanary:opencanary /etc/opencanaryd
```

4️⃣ Grant Python privileged-port access

```
sudo setcap 'cap_net_bind_service=+ep' "$(readlink -f /opt/opencanary/venv/bin/python)"
```

5️⃣ Install SysV service script

```
sudo nano /etc/init.d/opencanary
```

Paste the **final script** built (includes mkdir + runtime fix).

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:          opencanary
# Required-Start:    $network $remote_fs
# Required-Stop:     $network $remote_fs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: OpenCanary honeypot (Twisted)
# Description:       Starts OpenCanary using twistd and the opencanary.tac file
### END INIT INFO

NAME="opencanary"
VENV="/opt/opencanary/venv"
TWISTD="$VENV/bin/twistd"
PIDDIR="/var/run/opencanary"
PIDFILE="$PIDDIR/opencanary.pid"
LOGFILE="/var/log/opencanary.log"

# Resolve TAC path from installed package; fallback to source checkout
TAC="$($VENV/bin/python - <<'PY'
import os
try:
    import opencanary
    p = os.path.join(os.path.dirname(opencanary.__file__), "opencanary.tac")
    print(p if os.path.isfile(p) else "")
except Exception:
    print("")
PY
)"
[ -z "$TAC" ] && TAC="/opt/opencanary-src/bin/opencanary.tac"

# Arguments to twistd: use our TAC, log file, and where to write the pid
DAEMON_OPTS="-y $TAC -l $LOGFILE --pidfile $PIDFILE"
```

```
do_start() {
    if [ ! -x "$TWISTD" ]; then
        echo "twistd not found at $TWISTD"; return 1
    fi
    if [ ! -f "$TAC" ]; then
        echo "TAC not found at $TAC"; return 1
    fi

    echo "Starting $NAME..."

    # Ensure runtime dirs/log exist on every boot (/var/run is tmpfs)
    mkdir -p "$PIDDIR"
    chown opencanary:opencanary "$PIDDIR"
    touch "$LOGFILE"
    chown opencanary:opencanary "$LOGFILE"

    # Start in background; let twistd create the pidfile (no --make-pidfile here)
    start-stop-daemon --start --oknodo --background \
      --pidfile "$PIDFILE" --chuid opencanary --exec "$TWISTD" -- $DAEMON_OPTS
}

do_stop() {
    echo "Stopping $NAME..."
    if [ -f "$PIDFILE" ]; then
        start-stop-daemon --stop --oknodo --pidfile "$PIDFILE"
        rm -f "$PIDFILE"
    else
        # Fallback if pidfile is missing
        pkill -f "twistd .*opencanary.tac" 2>/dev/null || true
    fi
}

do_status() {
    if [ -f "$PIDFILE" ] && ps -p "$(cat "$PIDFILE" 2>/dev/null)" >/dev/null 2>&1; then
        echo "$NAME running (pid $(cat "$PIDFILE"))"; return 0
    fi
    pgrep -f "twistd .*opencanary.tac" >/dev/null 2>&1 && { echo "$NAME running"; return 0; }
    echo "$NAME not running"; return 3
}

case "$1" in
  start)   do_start ;;
  stop)    do_stop ;;
  restart) do_stop; sleep 1; do_start ;;
  status)  do_status ;;
  *) echo "Usage: /etc/init.d/$NAME {start|stop|restart|status}"; exit 1 ;;
esac
exit $?
```

After pasting and saving:

```
sudo dos2unix /etc/init.d/opencanary 2>/dev/null || true
sudo chmod +x /etc/init.d/opencanary
sudo update-rc.d -f opencanary remove
sudo update-rc.d opencanary defaults
```

6️⃣Create service user and log dirs

```
sudo useradd -r -s /usr/sbin/nologin opencanary 2>/dev/null || true
sudo mkdir -p /var/run/opencanary /var/log
sudo touch /var/log/opencanary.log
sudo chown -R opencanary:opencanary /opt/opencanary /etc/opencanaryd /var/run/opencanary /var/log/opencanary.log
```

7️⃣Start & verify

```
sudo service opencanary start
sudo service opencanary status
sudo tail -f /var/log/opencanary.log
sudo ss -tulpen | grep -E ':21|:22|:80' || true
```

✅ Should show twistd listening on 21, 22, 80.

8️⃣Firewall & network

```
sudo ufw allow 21/tcp
sudo ufw allow 22/tcp
sudo ufw allow 80/tcp
sudo ufw enable
sudo ufw status
```

9️⃣Auto-start check

```
sudo reboot
sudo service opencanary status
```

Should show `opencanary running (pid XXXX)` automatically.

Syslog to Kali Purple (later)

Edit `/etc/opencanaryd/opencanary.conf`:

This is using “nano”

```
"syslog": { "enabled": true, "host": "KALI_IP", "port": 514 }
```

1️⃣1️⃣ Testing your honeypot

From your victim Windows device:

```
curl http://<HONEYPOT_IP>/
telnet <HONEYPOT_IP> 21
ssh <HONEYPOT_IP> -p 22
```

Back on MX:

```
sudo tail -f /var/log/opencanary.log
```

1️⃣2️⃣ Log rotation (optional but recommended)

```
sudo tee /etc/logrotate.d/opencanary >/dev/null <<'EOF'
/var/log/opencanary.log {
  weekly
  rotate 8
  compress
  missingok
  notifempty
  create 0640 opencanary opencanary
  postrotate
    /usr/sbin/service opencanary restart > /dev/null 2>&1 || true
  endscript
}
EOF
```

Enjoy!