Cyber Physical Systems Security 12th Project

Participants:

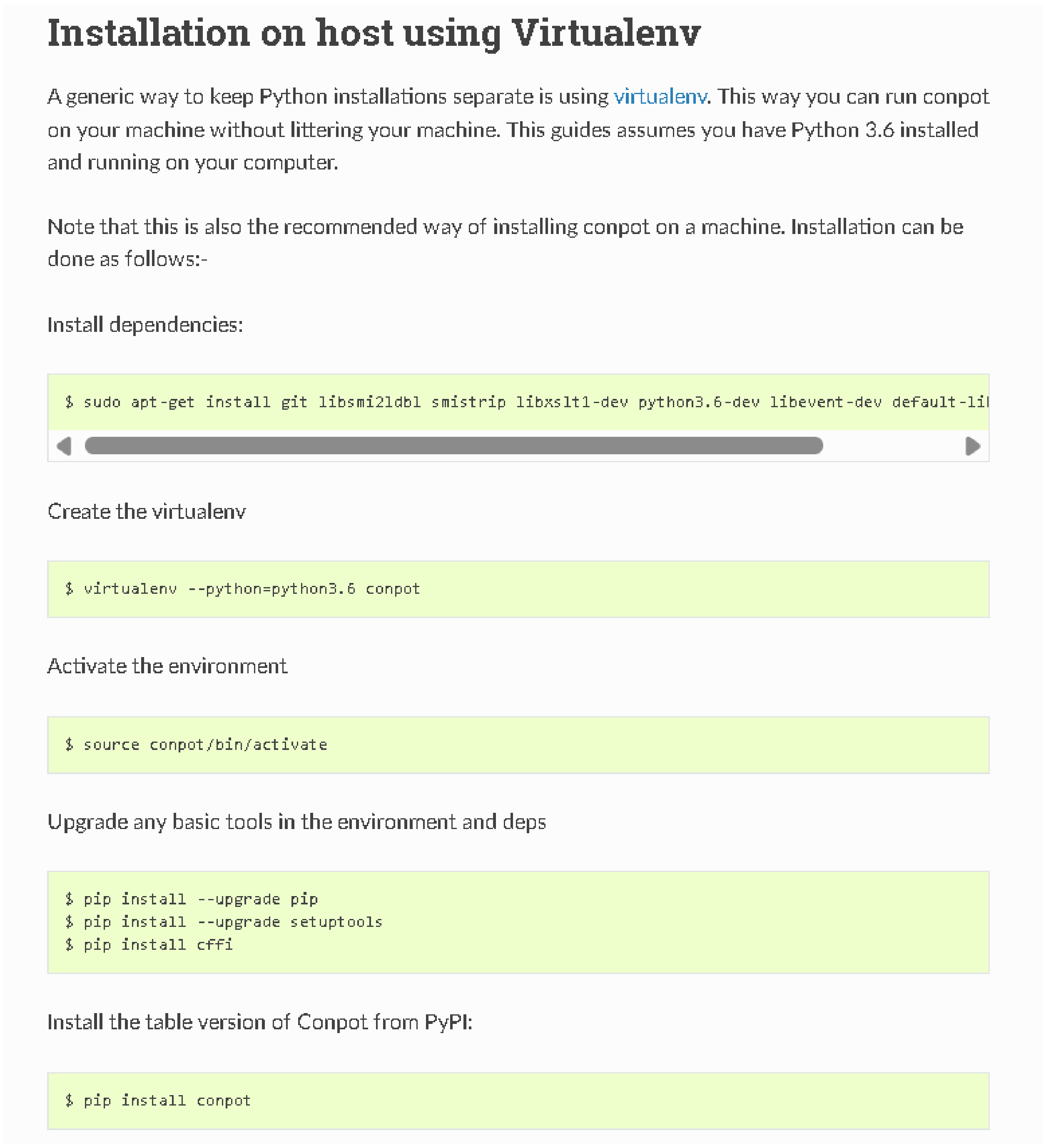
- Mohamed Mostafa Sayed Saber Ali	**221027679**
- Mohamed Hany El Kordy	**221027552**
- Basel Hany Abbas	**221027635**
- Mohamed Basem	**221027617**
- Mohamed Mostafa Abd El Baset	**221027785**
- Omar Hesham	**221027739**

<u>Introduction</u>

- **Industrial Control Systems (ICS)** are essential in critical infrastructure such as **energy, water treatment, transportation, and manufacturing.** Because these systems are increasingly connected to IT networks, **they are often exposed to reconnaissance attempts and exploitation by attackers.** To study this behavior safely, security professionals use **ICS honeypots, which imitate real industrial devices and allow observation of attacker activity without risk to actual systems.**
- In this experiment, a lightweight **ICS honeypot, <u>Conpot</u>,** was installed and configured using its **<u>default template</u>**. **The goal was to simulate a realistic control system environment, expose ICS-related services, and observe how external tools such as <u>Nmap</u> interact with it.** **<u>Wireshark</u>** was used to **capture the network traffic generated during these interactions. This report documents the exposed services, scans performed, logs collected, and insights gained about ICS reconnaissance behavior.**

Honeypot Setup and Configuration

- The honeypot used in this experiment, **Conpot**, was installed by following the official documentation provided at: Press Here



Installation on host using Virtualenv

A generic way to keep Python installations separate is using virtualenv. This way you can run conpot on your machine without littering your machine. This guides assumes you have Python 3.6 installed and running on your computer.

Note that this is also the recommended way of installing conpot on a machine. Installation can be done as follows:-

Install dependencies:

```
$ sudo apt-get install git libsmi2ldbl smistrip libxslt1-dev python3.6-dev libevent-dev default-lil
```

Create the virtualenv

```
$ virtualenv --python=python3.6 conpot
```

Activate the environment

```
$ source conpot/bin/activate
```

Upgrade any basic tools in the environment and deps

```
$ pip install --upgrade pip
$ pip install --upgrade setuptools
$ pip install cffi
```

Install the table version of Conpot from PyPI:

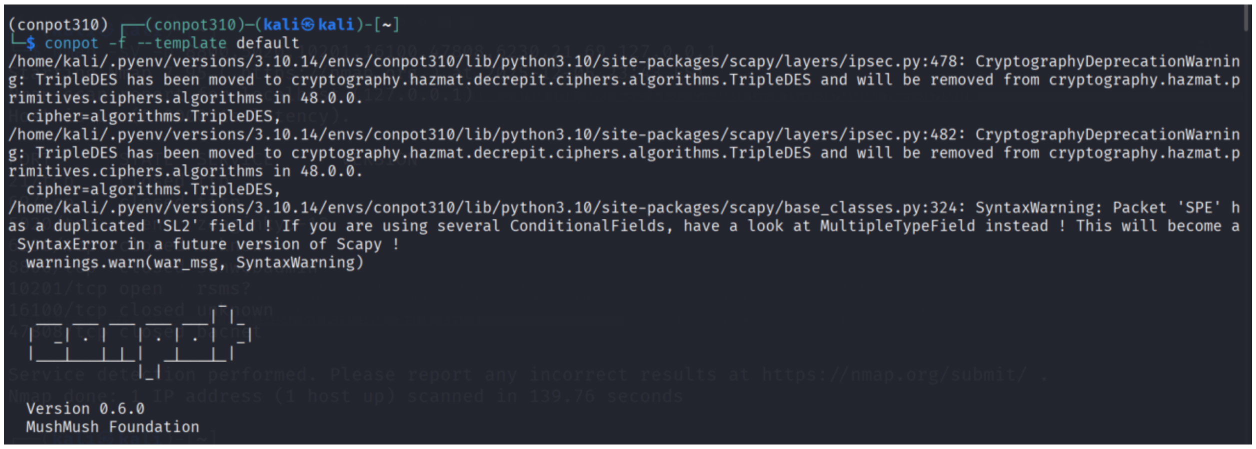
```
$ pip install conpot
```

- From the available installation methods, the **virtual environment (virtualenv)** setup was selected instead of **Docker**. Although **Conpot**

provides a Docker image, it is known to occasionally experience compatibility issues on certain systems, especially when ICS-related network bindings or low-level protocol ports are involved. **Using a dedicated Python virtual environment offered better stability and more control over dependencies.**

- A clean Python 3.10 environment was created, and Conpot was installed inside this isolated virtual environment. Once installation was completed, the honeypot was launched with **(as the following screenshot)**:

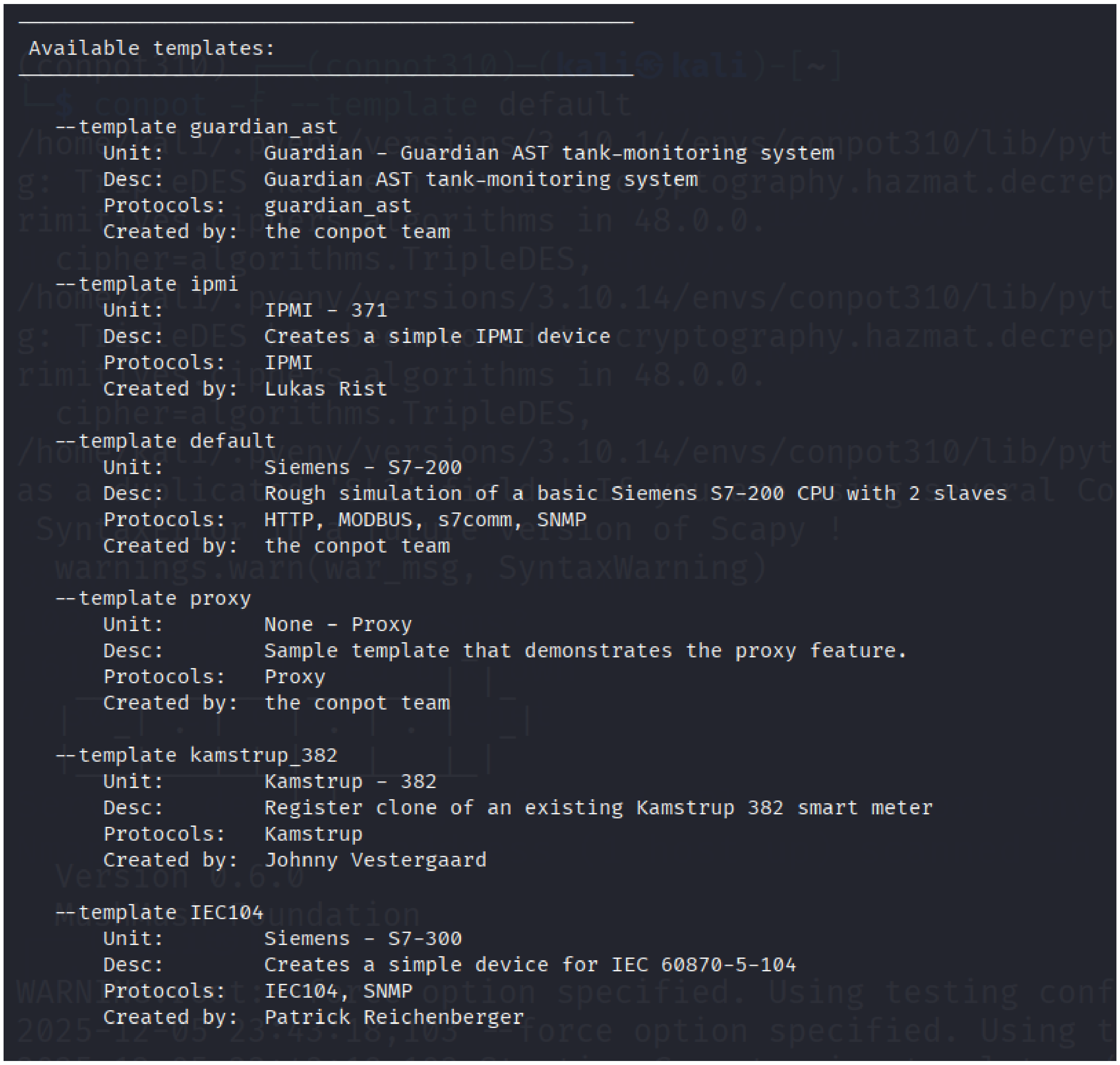
conpot -f --template default



```
(conpot310) ┌──(conpot310)─(kali㉿kali)-[~]
└─$ conpot -f --template default
/home/kali/.pyenv/versions/3.10.14/envs/conpot310/lib/python3.10/site-packages/scapy/layers/ipsec.py:478: CryptographyDeprecationWarnin
g: TripleDES has been moved to cryptography.hazmat.decrepit.ciphers.algorithms.TripleDES and will be removed from cryptography.hazmat.p
rimitives.ciphers.algorithms in 48.0.0.
  cipher=algorithms.TripleDES,
/home/kali/.pyenv/versions/3.10.14/envs/conpot310/lib/python3.10/site-packages/scapy/layers/ipsec.py:482: CryptographyDeprecationWarnin
g: TripleDES has been moved to cryptography.hazmat.decrepit.ciphers.algorithms.TripleDES and will be removed from cryptography.hazmat.p
rimitives.ciphers.algorithms in 48.0.0.
  cipher=algorithms.TripleDES,
/home/kali/.pyenv/versions/3.10.14/envs/conpot310/lib/python3.10/site-packages/scapy/base_classes.py:324: SyntaxWarning: Packet 'SPE' h
as a duplicated 'SL2' field ! If you are using several ConditionalFields, have a look at MultipleTypeField instead ! This will become a
 SyntaxError in a future version of Scapy !
  warnings.warn(war_msg, SyntaxWarning)

                       _
   ___ ___ ___ ___ ___| |_
  |  _| . |   | . | . |  _|
  |___|___|_|_|  _|___|_|
              |_|

  Version 0.6.0
  MushMush Foundation
```

- The **warnings** shown in this image are normal and not **fatal errors.**
- This command starts Conpot in the foreground using the **default template**, which simulates a **basic industrial controller**. When **Conpot** initializes, it builds a virtual filesystem, loads ICS protocol modules, and begins listening on multiple network interfaces. The default profile enables several industrial communication services such as **Modbus, S7Comm, BACnet, SNMP, ENIP, FTP,** and others, providing a realistic surface that resembles a live industrial automation device.

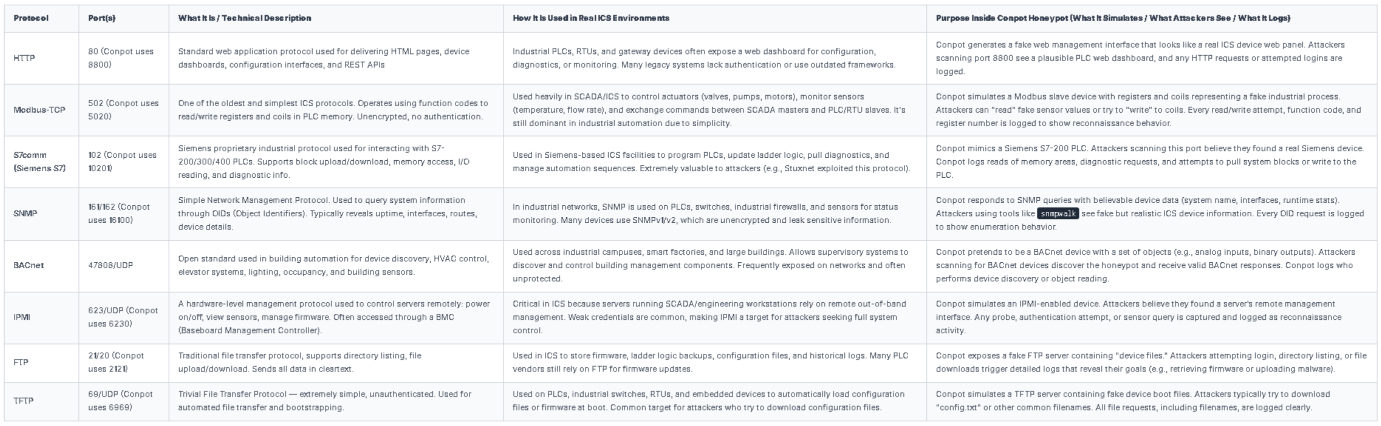
- **Another available templates if needed:**



```
 Available templates:

   --template guardian_ast
       Unit:         Guardian - Guardian AST tank-monitoring system
       Desc:         Guardian AST tank-monitoring system
       Protocols:    guardian_ast
       Created by:   the conpot team

   --template ipmi
       Unit:         IPMI - 371
       Desc:         Creates a simple IPMI device
       Protocols:    IPMI
       Created by:   Lukas Rist

   --template default
       Unit:         Siemens - S7-200
       Desc:         Rough simulation of a basic Siemens S7-200 CPU with 2 slaves
       Protocols:    HTTP, MODBUS, s7comm, SNMP
       Created by:   the conpot team

   --template proxy
       Unit:         None - Proxy
       Desc:         Sample template that demonstrates the proxy feature.
       Protocols:    Proxy
       Created by:   the conpot team

   --template kamstrup_382
       Unit:         Kamstrup - 382
       Desc:         Register clone of an existing Kamstrup 382 smart meter
       Protocols:    Kamstrup
       Created by:   Johnny Vestergaard

   --template IEC104
       Unit:         Siemens - S7-300
       Desc:         Creates a simple device for IEC 60870-5-104
       Protocols:    IEC104, SNMP
       Created by:   Patrick Reichenberger
```

<u>Services and Ports Exposed by Conpot</u>

- Below is a list of the industrial and auxiliary services that Conpot exposes based on your startup output:

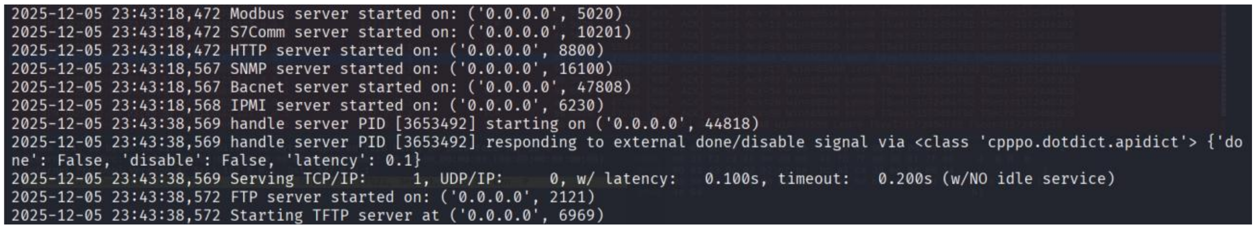


Protocol	Port(s)	What It Is / Technical Description	How It Is Used in Real ICS Environments	Purpose Inside Conpot Honeypot (What It Simulates / What Attackers See / What It Logs)
HTTP	80 (Conpot uses 8800)	Standard web application protocol used for delivering HTML pages, device dashboards, configuration interfaces, and REST APIs	Industrial PLCs, RTUs, and gateway devices often expose a web dashboard for configuration, diagnostics, or monitoring. Many legacy systems lack authentication or use outdated frameworks.	Conpot generates a fake web management interface that looks like a real ICS device web panel. Attackers scanning port 8800 see a plausible PLC web dashboard, and any HTTP requests or attempted logins are logged.
Modbus-TCP	502 (Conpot uses 5020)	One of the oldest and simplest ICS protocols. Operates using function codes to read/write registers and coils in PLC memory. Unencrypted, no authentication.	Used heavily in SCADA/ICS to control actuators (valves, pumps, motors), monitor sensors (temperature, flow rate), and exchange commands between SCADA masters and PLC/RTU slaves. It's still dominant in industrial automation due to simplicity.	Conpot simulates a Modbus slave device with registers and coils representing a fake industrial process. Attackers can "read" fake sensor values or try to "write" to coils. Every read/write attempt, function code, and register number is logged to show reconnaissance behavior.
S7comm (Siemens S7)	102 (Conpot uses 10201)	Siemens proprietary industrial protocol used for interacting with S7-200/300/400 PLCs. Supports block upload/download, memory access, I/O reading, and diagnostic info.	Used in Siemens-based ICS facilities to program PLCs, update ladder logic, pull diagnostics, and manage automation sequences. Extremely valuable to attackers (e.g., Stuxnet exploited this protocol).	Conpot mimics a Siemens S7-200 PLC. Attackers scanning this port believe they found a real Siemens device. Conpot logs reads of memory areas, diagnostic requests, and attempts to pull system blocks or write to the PLC.
SNMP	161/162 (Conpot uses 16100)	Simple Network Management Protocol. Used to query system information through OIDs (Object Identifiers). Typically reveals uptime, interfaces, routes, device details.	In industrial networks, SNMP is used on PLCs, switches, industrial firewalls, and sensors for status monitoring. Many devices use SNMPv1/v2, which are unencrypted and leak sensitive information.	Conpot responds to SNMP queries with believable device data (system name, interfaces, runtime stats). Attackers using tools like `snmpwalk` see fake but realistic ICS device information. Every OID request is logged to show enumeration behavior.
BACnet	47808/UDP	Open standard used in building automation for device discovery, HVAC control, elevator systems, lighting, occupancy, and building sensors.	Used across industrial campuses, smart factories, and large buildings. Allows supervisory systems to discover and control building management components. Frequently exposed on networks and often unprotected.	Conpot pretends to be a BACnet device with a set of objects (e.g., analog inputs, binary outputs). Attackers scanning for BACnet devices discover the honeypot and receive valid BACnet responses. Conpot logs who performs device discovery or object reading.
IPMI	623/UDP (Conpot uses 6230)	A hardware-level management protocol used to control servers remotely: power on/off, view sensors, manage firmware. Often accessed through a BMC (Baseboard Management Controller).	Critical in ICS because servers running SCADA/engineering workstations rely on remote out-of-band management. Weak credentials are common, making IPMI a target for attackers seeking full system control.	Conpot simulates an IPMI-enabled device. Attackers believe they found a server's remote management interface. Any probe, authentication attempt, or sensor query is captured and logged as reconnaissance activity.
FTP	21/20 (Conpot uses 2121)	Traditional file transfer protocol, supports directory listing, file upload/download. Sends all data in cleartext.	Used in ICS to store firmware, ladder logic backups, configuration files, and historical logs. Many PLC vendors still rely on FTP for firmware updates.	Conpot exposes a fake FTP server containing "device files." Attackers attempting login, directory listing, or file downloads trigger detailed logs that reveal their goals (e.g., retrieving firmware or uploading malware).
TFTP	69/UDP (Conpot uses 6969)	Trivial File Transfer Protocol — extremely simple, unauthenticated. Used for automated file transfer and bootstrapping.	Used on PLCs, industrial switches, RTUs, and embedded devices to automatically load configuration files or firmware at boot. Common target for attackers who try to download configuration files.	Conpot simulates a TFTP server containing fake device boot files. Attackers typically try to download "config.txt" or other common filenames. All file requests, including filenames, are logged clearly.

- These ports mimic what would be found on **ICS PLCs or building management controllers. An attacker performing reconnaissance would immediately recognize this as a potentially valuable industrial target.**

<u>Baseline Honeypot Output (Before Any Scanning)</u>

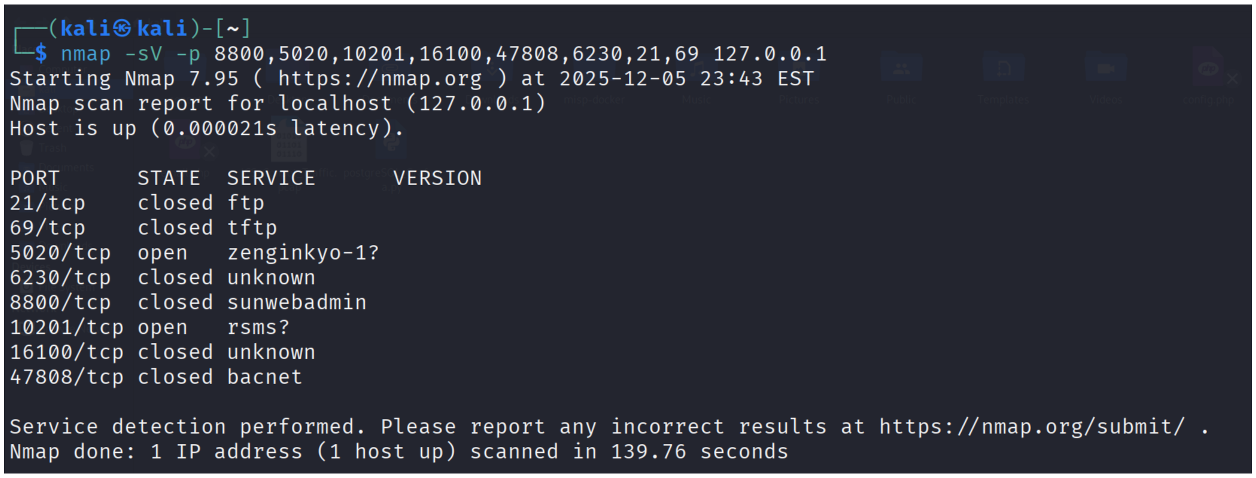
- **Before running Nmap or any scanner, Conpot logs normal startup messages and internal interactions.**
- Typical output observed:
  - **Initialization of Modbus, S7comm, HTTP, SNMP, BACnet, IPMI, ENIP, FTP, TFTP**
  - Creation of virtual filesystem directories
  - Detection of external IP address
  - No external sessions except internal Conpot checks
- **Example excerpt:**



```
2025-12-05 23:43:18,472 Modbus server started on: ('0.0.0.0', 5020)
2025-12-05 23:43:18,472 S7Comm server started on: ('0.0.0.0', 10201)
2025-12-05 23:43:18,472 HTTP server started on: ('0.0.0.0', 8800)
2025-12-05 23:43:18,567 SNMP server started on: ('0.0.0.0', 16100)
2025-12-05 23:43:18,567 Bacnet server started on: ('0.0.0.0', 47808)
2025-12-05 23:43:18,568 IPMI server started on: ('0.0.0.0', 6230)
2025-12-05 23:43:38,569 handle server PID [3653492] starting on ('0.0.0.0', 44818)
2025-12-05 23:43:38,569 handle server PID [3653492] responding to external done/disable signal via <class 'cpppo.dotdict.apidict'> {'do
ne': False, 'disable': False, 'latency': 0.1}
2025-12-05 23:43:38,569 Serving TCP/IP:     1, UDP/IP:     0, w/ latency:   0.100s, timeout:   0.200s (w/NO idle service)
2025-12-05 23:43:38,572 FTP server started on: ('0.0.0.0', 2121)
2025-12-05 23:43:38,572 Starting TFTP server at ('0.0.0.0', 6969)
```

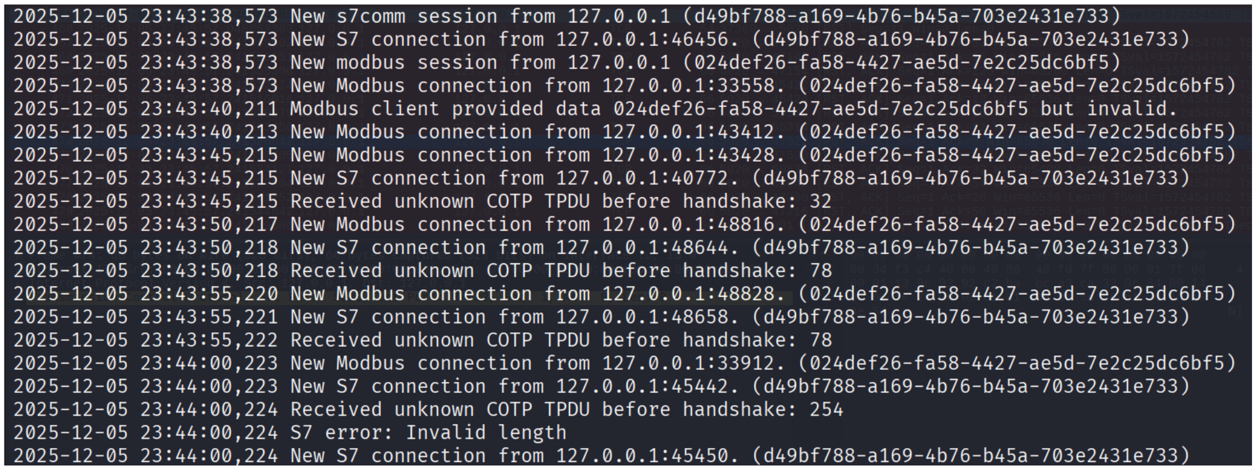
Interaction After Running Nmap (Scanner Activity)

- **After you ran an Nmap scan, Conpot generated numerous logs showing connection attempts, malformed packets, and protocol negotiation failures**, typical when generic scanners probe ICS services they do not fully understand.

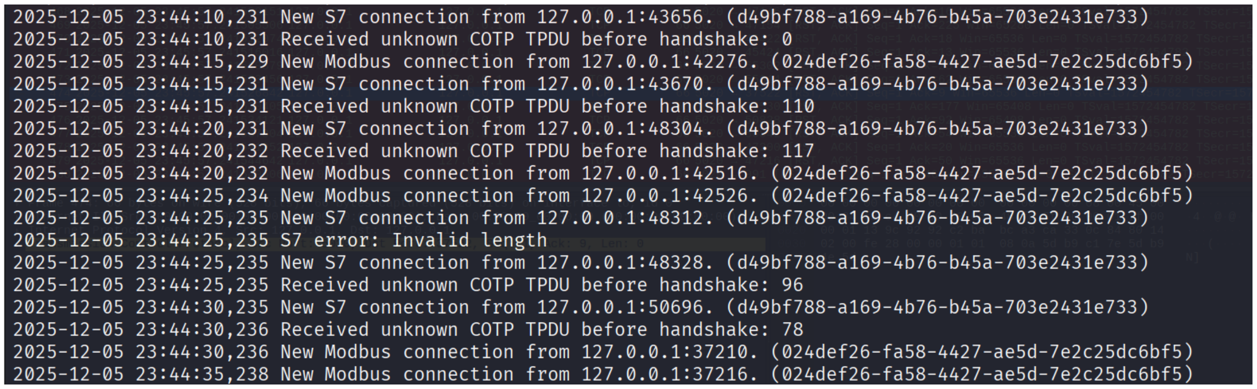


```
┌──(kali㉿kali)-[~]
└─$ nmap -sV -p 8800,5020,10201,16100,47808,6230,21,69 127.0.0.1
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-05 23:43 EST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000021s latency).

PORT      STATE  SERVICE      VERSION
21/tcp    closed ftp
69/tcp    closed tftp
5020/tcp  open   zenginkyo-1?
6230/tcp  closed unknown
8800/tcp  closed sunwebadmin
10201/tcp open   rsms?
16100/tcp closed unknown
47808/tcp closed bacnet

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 139.76 seconds
```

- **Evidence of Scanning** (Repeated new Modbus connections, Continuous S7Comm connections, and ENIP activity (scanner probing CIP/EthernetIP port**)**:



```
2025-12-05 23:43:38,573 New s7comm session from 127.0.0.1 (d49bf788-a169-4b76-b45a-703e2431e733)
2025-12-05 23:43:38,573 New S7 connection from 127.0.0.1:46456. (d49bf788-a169-4b76-b45a-703e2431e733)
2025-12-05 23:43:38,573 New modbus session from 127.0.0.1 (024def26-fa58-4427-ae5d-7e2c25dc6bf5)
2025-12-05 23:43:38,573 New Modbus connection from 127.0.0.1:33558. (024def26-fa58-4427-ae5d-7e2c25dc6bf5)
2025-12-05 23:43:40,211 Modbus client provided data 024def26-fa58-4427-ae5d-7e2c25dc6bf5 but invalid.
2025-12-05 23:43:40,213 New Modbus connection from 127.0.0.1:43412. (024def26-fa58-4427-ae5d-7e2c25dc6bf5)
2025-12-05 23:43:45,215 New Modbus connection from 127.0.0.1:43428. (024def26-fa58-4427-ae5d-7e2c25dc6bf5)
2025-12-05 23:43:45,215 New S7 connection from 127.0.0.1:40772. (d49bf788-a169-4b76-b45a-703e2431e733)
2025-12-05 23:43:45,215 Received unknown COTP TPDU before handshake: 32
2025-12-05 23:43:50,217 New Modbus connection from 127.0.0.1:48816. (024def26-fa58-4427-ae5d-7e2c25dc6bf5)
2025-12-05 23:43:50,218 New S7 connection from 127.0.0.1:48644. (d49bf788-a169-4b76-b45a-703e2431e733)
2025-12-05 23:43:50,218 Received unknown COTP TPDU before handshake: 78
2025-12-05 23:43:55,220 New Modbus connection from 127.0.0.1:48828. (024def26-fa58-4427-ae5d-7e2c25dc6bf5)
2025-12-05 23:43:55,221 New S7 connection from 127.0.0.1:48658. (d49bf788-a169-4b76-b45a-703e2431e733)
2025-12-05 23:43:55,222 Received unknown COTP TPDU before handshake: 78
2025-12-05 23:44:00,223 New Modbus connection from 127.0.0.1:33912. (024def26-fa58-4427-ae5d-7e2c25dc6bf5)
2025-12-05 23:44:00,223 New S7 connection from 127.0.0.1:45442. (d49bf788-a169-4b76-b45a-703e2431e733)
2025-12-05 23:44:00,224 Received unknown COTP TPDU before handshake: 254
2025-12-05 23:44:00,224 S7 error: Invalid length
2025-12-05 23:44:00,224 New S7 connection from 127.0.0.1:45450. (d49bf788-a169-4b76-b45a-703e2431e733)
```



```
2025-12-05 23:44:10,231 New S7 connection from 127.0.0.1:43656. (d49bf788-a169-4b76-b45a-703e2431e733)
2025-12-05 23:44:10,231 Received unknown COTP TPDU before handshake: 0
2025-12-05 23:44:15,229 New Modbus connection from 127.0.0.1:42276. (024def26-fa58-4427-ae5d-7e2c25dc6bf5)
2025-12-05 23:44:15,231 New S7 connection from 127.0.0.1:43670. (d49bf788-a169-4b76-b45a-703e2431e733)
2025-12-05 23:44:15,231 Received unknown COTP TPDU before handshake: 110
2025-12-05 23:44:20,231 New S7 connection from 127.0.0.1:48304. (d49bf788-a169-4b76-b45a-703e2431e733)
2025-12-05 23:44:20,232 Received unknown COTP TPDU before handshake: 117
2025-12-05 23:44:20,232 New Modbus connection from 127.0.0.1:42516. (024def26-fa58-4427-ae5d-7e2c25dc6bf5)
2025-12-05 23:44:25,234 New Modbus connection from 127.0.0.1:42526. (024def26-fa58-4427-ae5d-7e2c25dc6bf5)
2025-12-05 23:44:25,235 New S7 connection from 127.0.0.1:48312. (d49bf788-a169-4b76-b45a-703e2431e733)
2025-12-05 23:44:25,235 S7 error: Invalid length
2025-12-05 23:44:25,235 New S7 connection from 127.0.0.1:48328. (d49bf788-a169-4b76-b45a-703e2431e733)
2025-12-05 23:44:25,235 Received unknown COTP TPDU before handshake: 96
2025-12-05 23:44:30,235 New S7 connection from 127.0.0.1:50696. (d49bf788-a169-4b76-b45a-703e2431e733)
2025-12-05 23:44:30,236 Received unknown COTP TPDU before handshake: 78
2025-12-05 23:44:30,236 New Modbus connection from 127.0.0.1:37210. (024def26-fa58-4427-ae5d-7e2c25dc6bf5)
2025-12-05 23:44:35,238 New Modbus connection from 127.0.0.1:37216. (024def26-fa58-4427-ae5d-7e2c25dc6bf5)
```

- These entries represent exactly what an **attacker's reconnaissance tooling looks like,** containing **connection IP address** and **session identifiers.**

Wireshark Traffic Capture (Observation Summary)

- To monitor how external tools interacted with the honeypot, a **packet capture was performed using Wireshark.** Because Conpot was running locally and being scanned using **nmap on 127.0.0.1, the capture interface was set to Loopback (lo).** To avoid noise from unrelated loopback traffic, **a targeted capture filter was applied that focuses only on the ports Conpot exposes:**

  tcp port 5020 or tcp port 10201 or tcp port 8800 or udp port 16100 or udp port 47808 or port 6230 or tcp port 2121 or udp port 6969
- This allowed us to **isolate and analyze only ICS-related activity generated by Conpot and the scanners interacting with it.**
- **What the Capture Contains (Breakdown by Protocol/Port):**
  - **Modbus/TCP (TCP 5020):** Wireshark shows connection attempts from Nmap (or other scanners) to port 5020, which Conpot uses as its Modbus simulation port.
  - The packets typically include:
    - SYN / SYN-ACK handshake
    - Short Modbus query attempts
    - Reset packets when malformed or unexpected data is sent

- **S7comm (TCP 10201):** This port receives a rapid sequence of handshake attempts as scanners probe Siemens S7 PLC behavior.
- In the capture, you'll see:
  - TPKT headers
  - COTP protocol initiation
  - Error responses from Conpot when packets are malformed
- These correlate directly with the S7Comm errors printed in your Conpot logs.
- **HTTP (TCP 8800):** The traffic includes:
  - Nmap service detection probes
  - HTTP GET requests or banner grabs
- Conpot returns fake web server responses from the template.
- **SNMP (UDP 16100):** The traffic includes:
  - SNMP GET/GET-NEXT requests originating from scanners
  - Conpot's fake SNMP replies based on the template
- The traffic appears as short UDP exchanges.
- **BACnet (UDP 47808):** Displays typical BACnet device discovery broadcasts or queries, Conpot replies with fake ICS BACnet point information.
- **IPMI (UDP 6230):** Shows unsolicited probes checking for an IPMI BMC (Baseboard Management Controller), Conpot responds with template-based IPMI packets.
- **FTP (TCP 2121):** The traffic includes:
  - TCP handshake
  - Anonymous login attempts from Nmap FTP scripts
  - Conpot's simulated FTP service banner
- **TFTP (UDP 6969):** The traffic includes:
  - TFTP Read/Write request attempts from Nmap
  - Conpot responding with minimal simulated TFTP headers

Analysis & Discussion

- **What type of system the honeypot pretends to be:**
    - The **Conpot default template** mimics a **generic industrial controller supporting several common ICS protocols.** It appears to attackers as:
        - **A Modbus-capable field device**
        - **A Siemens S7 PLC**
        - **A BACnet building controller**
        - **A CIP/ENIP industrial Ethernet node**
        - **A device with management services (FTP, SNMP, IPMI)**
    - To an attacker, this looks like a real piece of industrial hardware integrated into a production environment.
- **What an attacker would see:**
    - From an attacker's perspective:
        - **Multiple ICS-specific ports open**
        - **A web server**
        - **SNMP service suggesting networked management**
        - **Firmware/config transfer services (FTP/TFTP)**
        - **Responses indicating valid protocol headers (even if limited)**
    - This is usually enough to convince a recon bot or human attacker that the system is valuable.
- **What the logs reveal about typical reconnaissance behavior:**
    - Logs show:
        - **Rapid, repeated connections to different ICS ports**
        - **Attempted protocol negotiations**
        - **Invalid packet structures (scanners trying to guess the correct handshake)**
        - **Time-based repetition of scan cycles**
        - **Multiple transport-layer resets**
    - This matches real-world ICS reconnaissance where attackers:
        - **Identify exposed ICS protocols**
        - **Try to extract version info**
        - **Attempt default commands**
        - **Record device metadata if available**
    - **Conpot** perfectly captures these patterns.

- **<u>Why ICS honeypots are important for defense:</u>**
  - ICS honeypots serve several critical defensive purposes:
    - **<u>Early detection:</u> They identify scanning bots and targeted ICS attacks before real systems are touched.**
    - **<u>Threat intelligence:</u> Logs reveal attacker tools, scripts, and methods.**
    - **<u>Decoy value:</u> They waste attacker time and resources.**
    - **<u>Risk-free research:</u> Defenders can study malicious ICS activity without endangering real industrial processes.**
  - **In environments where safety and reliability are paramount, honeypots provide insight that cannot be gained directly from live systems.**

<u>Conclusion</u>

- This experiment successfully deployed an **ICS honeypot using Conpot, exposed multiple industrial protocols, and captured scanner interactions.** The **logs and Wireshark captures demonstrate typical reconnaissance behavior against ICS systems.** By observing these interactions in a controlled environment, defenders gain valuable insight into how attackers identify industrial devices and prepare for further exploitation.
- **ICS honeypots** remain an **essential tool** for **strengthening industrial cybersecurity** and improving **overall situational awareness.**

<u>Thank You</u>