Industrial Intrusion — OSINT 2 Walkthrough

🕵️ OSINT 2 — Confirming the Uplink Channel

Tags: #TryHackMe #OSINT #ThreatIntel #CTF #SubdomainEnumeration #Phishing #CyberSecurity #DigitalForensics

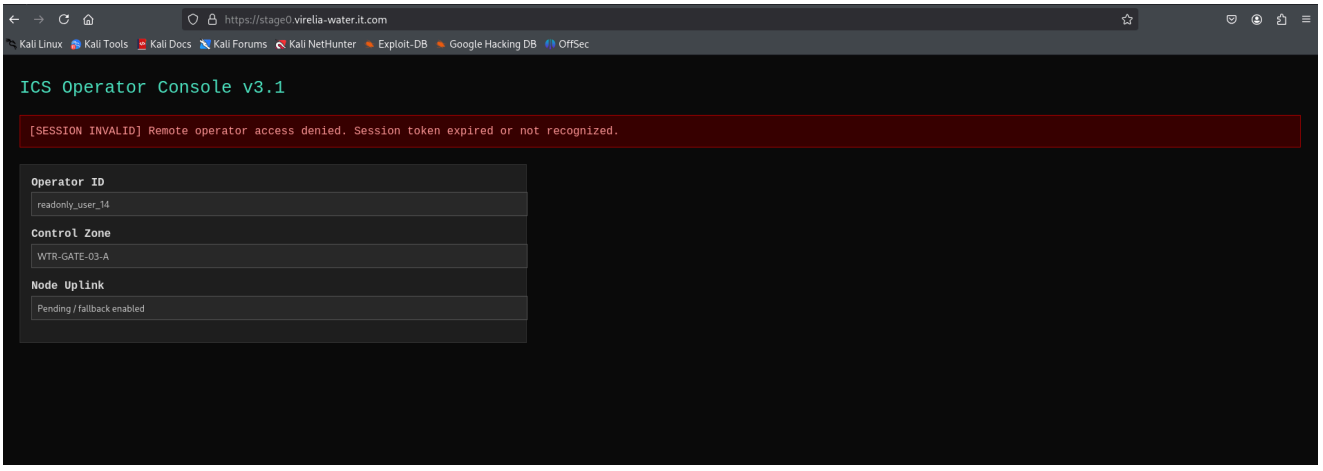
Scenario

> *“Great work on uncovering that suspicious subdomain, Hexline. However, your work here isn’t done yet — we believe there is more.”*

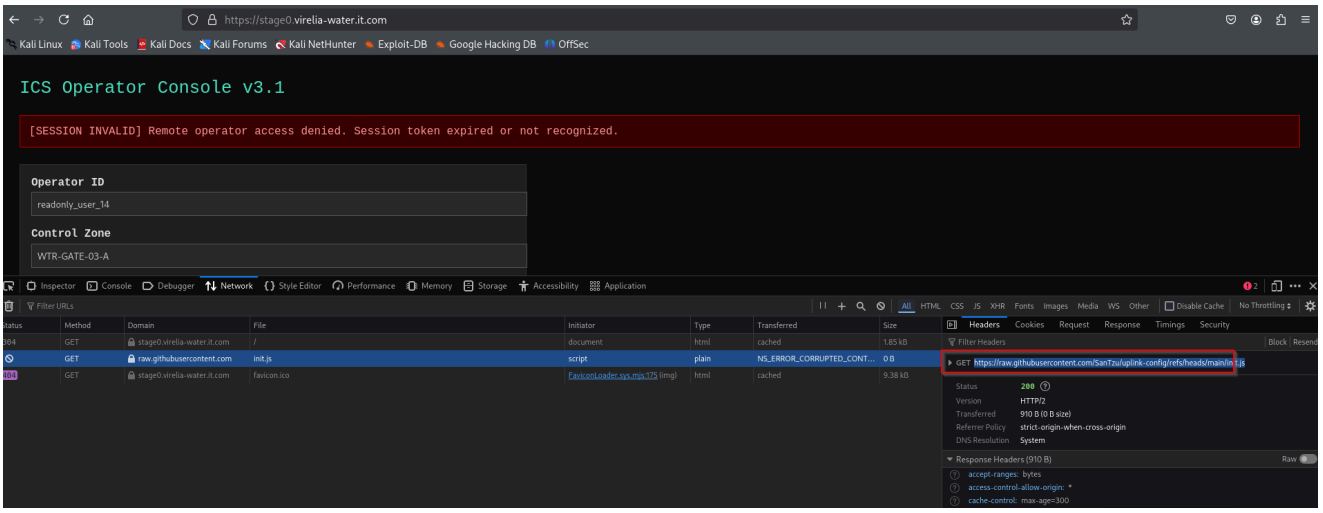
Objective: Your goal in this task was to **expand the investigation** from Task 5 by identifying additional infrastructure or connections linked to the original phishing campaign targeting **virelia-water.it.com**.

1. Approach:

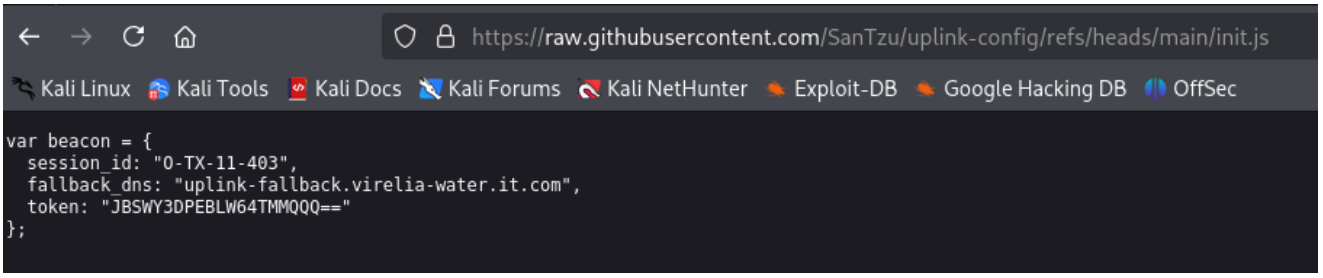
1. Accessing the ICS Operator Console: I navigated to the stage0.virelia-water.it.com subdomain, which hosted an ICS operator console. On load, I was greeted with an expired session token error. This could have been a dead end, but I knew that attackers often leave traces even after their access is invalidated. I dug deeper by inspecting the page source.



2. Examining HTTP Traffic: Using my browser’s developer tools, I inspected the network traffic and found a request for https://raw.githubusercontent.com/SanTzu/uplink-config/refs/heads/main/init.js. This file was hosted on GitHub and contained crucial information, including a session ID and a fallback DNS entry.



3. Extracting the Token: The value that caught my eye was a base64-encoded string under the token field. A quick decode of the string gave me the value "hello". Although this wasn't a flag, it was part of the attacker's token data that likely played a role in their environment setup. Decoded Value: hello



2. Investigating Uplink Channel – DNS Query Analysis

Continuing my investigation, I decided to analyze the DNS records associated with the potential fallback channel used by the attackers. I queried the **uplink-fallback.virelia-water.it.com** subdomain for TXT records. This revealed a base64-encoded string, which likely serves as part of the attacker's infrastructure for maintaining persistent communication or backdoor access.

```
dig txt uplink-fallback.virelia-water.it.com
```

The output returned the following base64 string:

```
eyJzZXNzaW9uIjoiVC1DTjEtMTcyIiwiZmxhZyI6IlRITXt1cGxpbmtfY2hhbm5lbF9jb25maXJt
ZWR9In0=
```

Decode from Base64 format

Simply enter your data then push the decode button.

eyJzZXNzaW9uIjoiVC1DTjEtMTcyIiwiZmxhZyI6IlRITXt1cGxpbmtfY2hhbm5lbF9jb25maXJtZWR9In0=

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Source character set.

Decode each line separately (useful for when you have multiple entries).

Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

DECODE Decodes your data into the area below.

{"session":"T-CN1-172","flag":"THM{uplink_channel_confirmed}"}

Tools Used:

- **GitHub** (For identifying hosted files)
- **Browser Developer Tools** (For inspecting network traffic)
- **Base64 Decoding** (For analyzing the attacker's token)
- **DNS Lookup** (For identifying fallback infrastructure)