FIELD NOTE / RED TEAMING

Getting Started with Covenant C2 for Red Teaming

Petruknisme · 21 Nov, 2021 · 8 min to read

Command and Control is part of Red Teaming tactic according to Mitre ATT&CK framework. C2 is very helpful for maintaining persistent access between attacker and compromised machine and easy to exfiltrate data. There are lot of C2 framework available to use from open-source version to paid-version like Empire (https://github.com/BC-SECURITY/Empire), SharpC2 (https://github.com/SharpC2/SharpC2), SilentTrinity (https://github.com/byt3bl33d3r/SILENTTRINITY), Metasploit (https://www.metasploit.com/), Covenant (https://github.com/cobbr/Covenant), Cobalt Strike (https://www.cobaltstrike.com/) (Paid), Merlin (https://github.com/Ne0nd0g/merlin), PoshC2 (https://github.com/nettitude/PoshC2), and many more.

In this guide, I will talk about basic Covenant C2 for Red Teaming.

Covenant is a.NET command and control framework that aims to highlight the attack surface of.NET, make the use of offensive.NET tradecraft easier, and serve as a collaborative command and control platform for red teamers.

Covenant is an ASP.NET Core, cross-platform application that includes a web-based interface that allows for multi-user collaboration.



https://github.com/cobbr/Covenant (https://github.com/cobbr/Covenant)

Architecture

Covenant has a client-server architecture that allows for multi-user collaboration. There are three main components of Covenant's architecture:

Licensed under CC BY-SA 4.0

- **Covenant** — Covenant is the server-side component of the client-server architecture. Covenant runs the command and control server hosted on infrastructure shared between operators. I will also frequently use the term “Covenant” to refer to the entire overarching project that includes all components of the architecture.
- **Elite** — Elite is the client-side component of the client-server architecture. Elite is a command-line interface that operators use to interact with the Covenant server to conduct operations.
- **Grunt** — A “Grunt” is the name of Covenant’s implant that is deployed to targets.

All three components of Covenant are written in C#. Covenant and Elite both target.NET Core and have docker support, while Grunt implants target the.NET framework.

Features

Covenant has several key features that make it useful and differentiate it from other command and control frameworks:

- **Intuitive Interface** — Covenant provides an intuitive web application to easily run a collaborative red team operation.
- **Multi-Platform** — Covenant targets.NET Core, which is multi-platform. This allows Covenant to run natively on Linux, MacOS, and Windows platforms. Additionally, Covenant has docker support, allowing it to run within a container on any system that has docker installed.
- **Multi-User** — Covenant supports multi-user collaboration. The ability to collaborate has become crucial for effective red team operations. Many users can interact with the same Covenant server and operate independently or collaboratively.
- **API Driven** — Covenant is driven by an API that enables multi-user collaboration and is easily extendible. Additionally, Covenant includes a Swagger UI that makes development and debugging easier and more convenient.
- **Listener Profiles** — Covenant supports listener “profiles” that control how the network communication between Grunt implants and Covenant listeners look on the wire.
- **Encrypted Key Exchange** — Covenant implements an encrypted key exchange between Grunt implants and Covenant listeners that is largely based on a similar exchange in the Empire project, in addition to optional SSL encryption. This achieves the cryptographic property of forward secrecy between Grunt implants.
- **Dynamic Compilation** — Covenant uses the Roslyn API for dynamic C# compilation. Every time a new Grunt is generated or a new task is assigned, the relevant code is recompiled and obfuscated with ConfuserEx, avoiding totally static payloads. Covenant reuses much of the compilation code from the SharpGen project, which I described in much more detail in a previous post.
- **Inline C# Execution** — Covenant borrows code and ideas from both the SharpGen and SharpShell projects to allow operators to execute C# one-liners on Grunt implants. This allows for similar

Licensed under CC BY-SA 4.0

functionality to that described in the SharpShell post, but allows the one-liners to be executed on remote implants.

- **Tracking Indicators** — Covenant tracks "indicators" throughout an operation, and summarizes them in the Indicators menu. This allows an operator to conduct actions that are tracked throughout an operation and easily summarize those actions to the blue team during or at the end of an assessment for deconfliction and educational purposes. This feature is still in it's infancy and still has room for improvement.
- **Developed in C#** — Personally, I enjoy developing in C#, which may not be a surprise for anyone that has read my latest blogs or tools. Not everyone might agree that development in C# is ideal, but hopefully everyone agrees that it is nice to have all components of the framework written in the same language. I've found it very convenient to write the server, client, and implant all in the same language. This may not be a true "feature", but hopefully it allows others to contribute to the project fairly easily.

Install Covenant

Covenant can be installed as docker container or manual install in linux machine. For easy to use, I'm highly recommended the docker version, so it already customized and ready to use.

Covenant is divided into two category: stable & dev version. Stable version use **SDK 3.1** and dev version using **SDK 5.0**

Stable version

```
$ git clone --recurse-submodules https://github.com/cobbr/Covenant
$ cd Covenant/Covenant
```

Dev version

```
$ git clone --recurse-submodules https://github.com/cobbr/Covenant -b dev
$ cd Covenant/Covenant
```

Licensed under CC BY-SA 4.0

Docker version

You need to build the image first and make sure docker is already installed on your machine. I will demonstrating all of the guide with Kali Linux, so adjust with your linux distro for the commands.

```
$ apt-get install docker && systemctl enable docker && systemctl start doc
$ cd Covenant/Covenant
$ sudo docker build -t covenant .
```

It will take lots of time to download and build the container. If you using dev version, you can edit dockerfile to use SDK 5.0 instead of SDK 3.1

Docker

```
┌──(kali㉿kali)-[~/Documents/tools/Covenant/Covenant]
└─$ sudo docker build -t covenant .
Sending build context to Docker daemon  99.21MB
Step 1/10 : FROM mcr.microsoft.com/dotnet/core/sdk:3.1 AS build
3.1: Pulling from dotnet/core/sdk
e22122b926a1: Pull complete
f29e09ae8373: Pull complete
e319e3daef68: Pull complete
e499244fe254: Pull complete
45167b1903e6: Pull complete
0df6347c1d39: Pull complete
98f6e4693223: Pull complete
Digest: sha256:572114cca6b20ffbe5e0d6788523a7de04b3e7432e731a8b1d6951bb085
Status: Downloaded newer image for mcr.microsoft.com/dotnet/core/sdk:3.1
 ---> a996740b0e4c
Step 2/10 : WORKDIR /app
 ---> Running in ec602e230a9b
Removing intermediate container ec602e230a9b
 ---> cfc0c7ef255d
Step 3/10 : COPY . ./
 ---> b5c16cb0cd58
Step 4/10 : RUN dotnet publish -c Release -o out
 ---> Running in c4dad333a0be
Microsoft (R) Build Engine version 16.7.2+b60ddb6f4 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.Determining proje
```

Licensed under CC BY-SA 4.0

```
  Restored /app/Covenant.csproj (in 8.32 sec).
  Covenant -> /app/bin/Release/netcoreapp3.1/Covenant.dll
  Covenant -> /app/bin/Release/netcoreapp3.1/Covenant.Views.dll
  Covenant -> /app/out/
Removing intermediate container c4dad333a0be
 ---> a7cb32121c62
Step 5/10 : FROM mcr.microsoft.com/dotnet/core/sdk:3.1 AS runtime
 ---> a996740b0e4c
Step 6/10 : WORKDIR /app
 ---> Using cache
 ---> cfc0c7ef255d
Step 7/10 : COPY --from=build /app/out .
 ---> b52d0f298c8f
Step 8/10 : COPY ./Data ./Data
 ---> 4693884e6716
Step 9/10 : EXPOSE 7443 80 443
 ---> Running in 729c5bf19810
Removing intermediate container 729c5bf19810
 ---> 8ba530274b0b
Step 10/10 : ENTRYPOINT ["dotnet", "Covenant.dll"]
 ---> Running in ecd6b017bae9
Removing intermediate container ecd6b017bae9
 ---> eff01ea5e6da
Successfully built eff01ea5e6da
Successfully tagged covenant:latest
```

Now we need to run the image container for using Covenant. From the wiki, it said that you need to input **`-v </absolute/path/to/Covenant/Covenant/Data>`** with your absolute path location, mine is **`/home/kali/Documents/tools/Covenant/Covenant/Data`**

```
sudo docker run -it -p 7443:7443 -p 80:80 -p 443:443 --name covenant -v /h
      Storing keys in a directory '/root/.aspnet/DataProtection-Keys' that
warn: Microsoft.EntityFrameworkCore.Model.Validation[10400]
      Sensitive data logging is enabled. Log entries and exception message
Covenant has started! Navigate to https://127.0.0.1:7443 in a browser
warn: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[35]
      No XML encryptor configured. Key {e1bd9426-088a-45d8-a4f5-cf4c42d5f7
```

Licensed under CC BY-SA 4.0

Linux Version

Covenant stable use dotnet-core 3.1 SDK, be sure to install the correct version. If you are using not using Kali Linux, you can download and install it directly https://dotnet.microsoft.com/download/dotnet-core/3.1 (https://dotnet.microsoft.com/download/dotnet-core/3.1) and https://dotnet.microsoft.com/download/dotnet-core/ (https://dotnet.microsoft.com/download/dotnet-core/3.1) 5.0 for dev version

Configure Kali Linux Repo

We need to install microsoft package repo to our Kali Linux, so we can install dotnet-core directly from apt-get. Please follow the step by step below:

```
┌──(kali㉿kali)-[~/Documents/tools/Covenant/Covenant]
└─$ wget https://packages.microsoft.com/config/debian/10/packages-microsof
--2021-03-25 21:09:51--  https://packages.microsoft.com/config/debian/10/p
Resolving packages.microsoft.com (packages.microsoft.com)... 20.188.102.6
Connecting to packages.microsoft.com (packages.microsoft.com)|20.188.102.6
HTTP request sent, awaiting response... 200 OK
Length: 3124 (3.1K) [application/octet-stream]
Saving to: ‘packages-microsoft-prod.deb’

packages-microsoft-prod.deb                    100%[====================

2021-03-25 21:09:52 (105 MB/s) - ‘packages-microsoft-prod.deb’ saved [3124


┌──(kali㉿kali)-[~/Documents/tools/Covenant/Covenant]
└─$ sudo dpkg -i packages-microsoft-prod.deb
[sudo] password for kali:
Selecting previously unselected package packages-microsoft-prod.
(Reading database ... 266566 files and directories currently installed.)
Preparing to unpack packages-microsoft-prod.deb ...
Unpacking packages-microsoft-prod (1.0-debian10.1) ...
Setting up packages-microsoft-prod (1.0-debian10.1) ...┌──(kali㉿kali)-[~/D
└─$ sudo apt-get update
Get:1 https://packages.microsoft.com/debian/10/prod buster InRelease [29.8
Hit:2 https://apt.releases.hashicorp.com buster InRelease
Get:3 https://packages.microsoft.com/debian/10/prod buster/main all Packag
Get:4 https://packages.microsoft.com/debian/10/prod buster/main amd64 Pack
Get:5 https://packages.microsoft.com/debian/10/prod buster/main armhf Pack
```

Licensed under CC BY-SA 4.0

```
29  Get:6 https://packages.microsoft.com/debian/10/prod buster/main arm64 Pack
30  Hit:7 https://linux.teamviewer.com/deb stable InRelease
31  Hit:8 http://kali.cs.nctu.edu.tw/kali kali-rolling InRelease
32  Fetched 144 kB in 2s (60.6 kB/s)
33  Reading package lists... Done
```

Install SDK

Stable use 3.1

```
1  $ sudo apt-get install dotnet-sdk-3.1
```

Dev use 5.0

```
1  $ sudo apt-get install dotnet-sdk-5.0
```

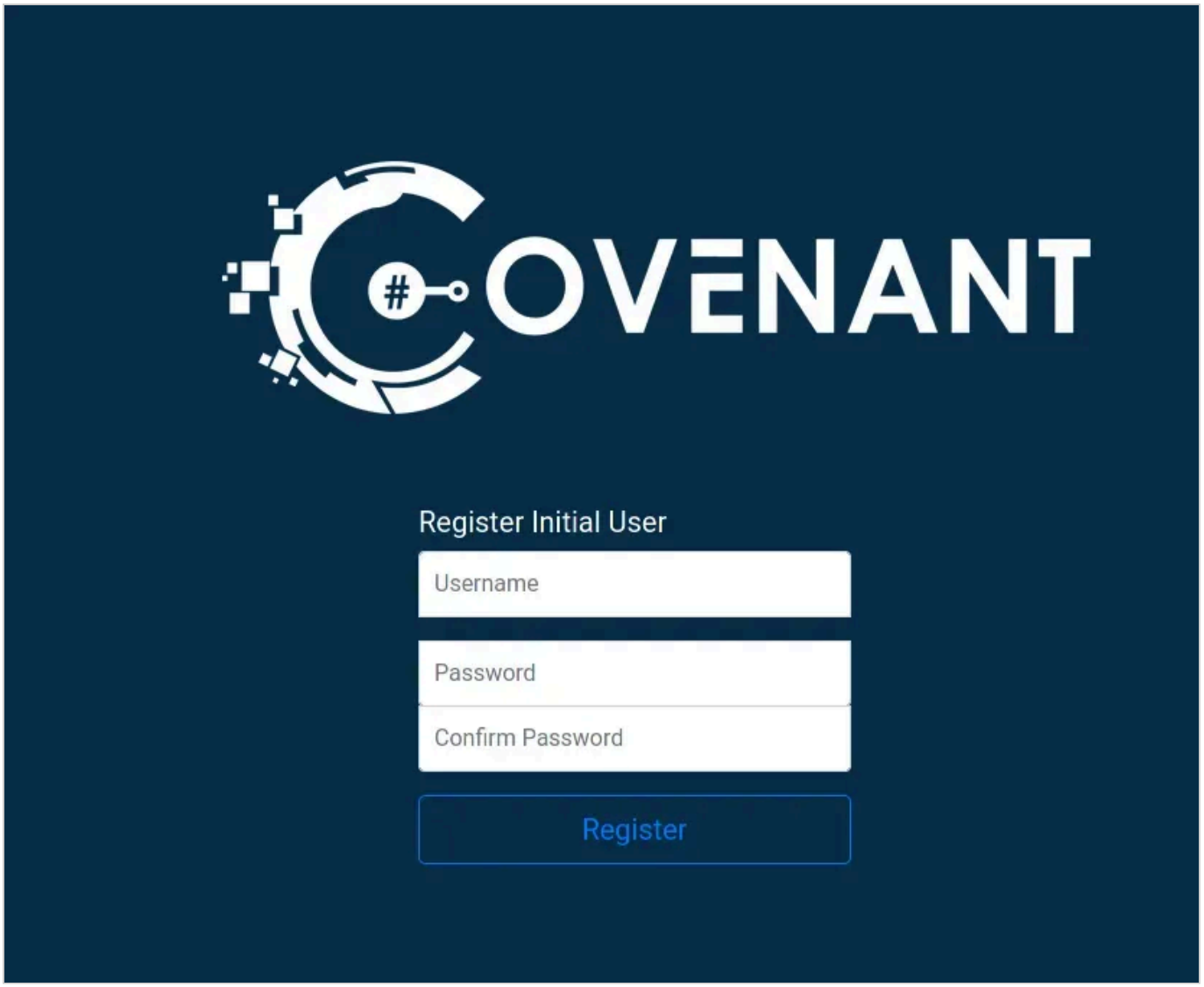
Install and running Covenant

Covenant

```
1  ┌──(kali㉿kali)-[~/Documents/tools/Covenant/Covenant]
2  └─$ dotnet runFound default JwtKey, replacing with auto-generated key...
3  warn: Microsoft.EntityFrameworkCore.Model.Validation[10400]
4        Sensitive data logging is enabled. Log entries and exception message
5  WARNING: Running Covenant non-elevated. You may not have permission to sta
6  Covenant has started! Navigate to https://127.0.0.1:7443 in a browser
7  Creating cert...
8  warn: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[35]
9        No XML encryptor configured. Key {1b710c70-528c-45b8-b678-ff39c40953
```

Covenant has started, we can navigate to https://<host-ip>:7443.

Licensed under CC BY-SA 4.0

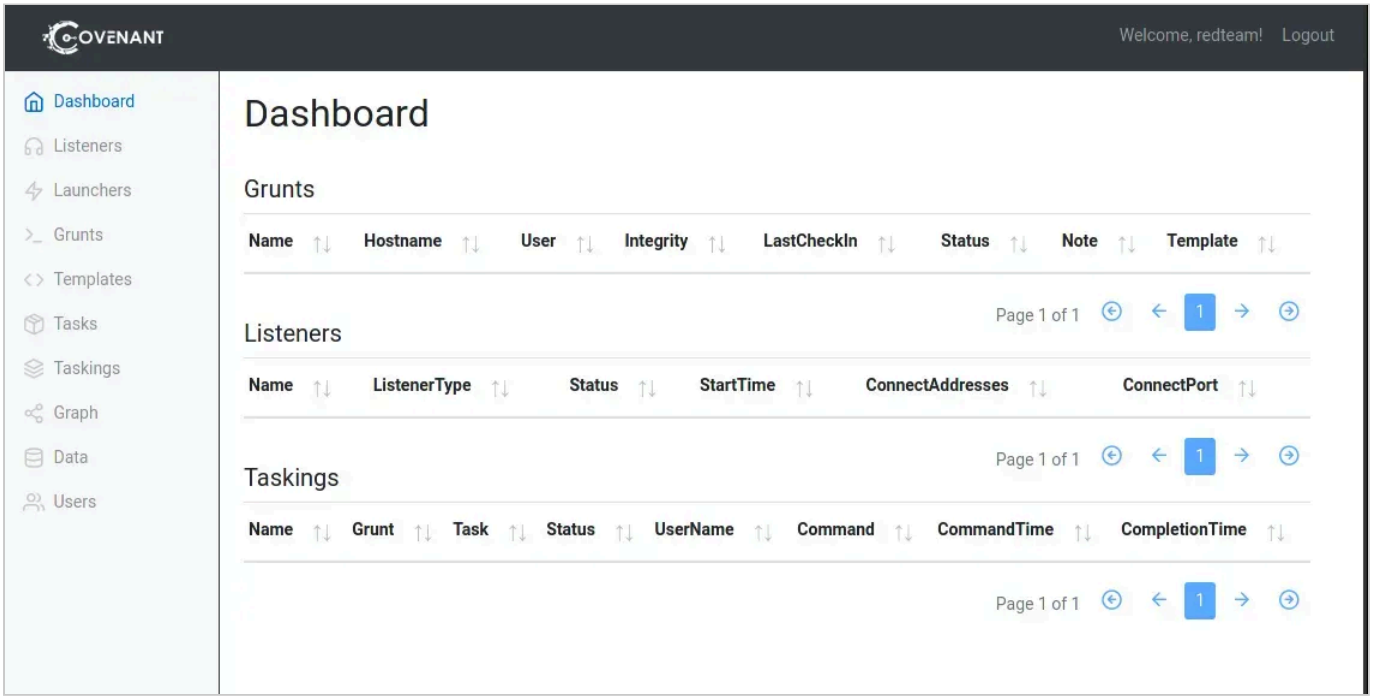


Covenant Welcome Page

Licensed under CC BY-SA 4.0



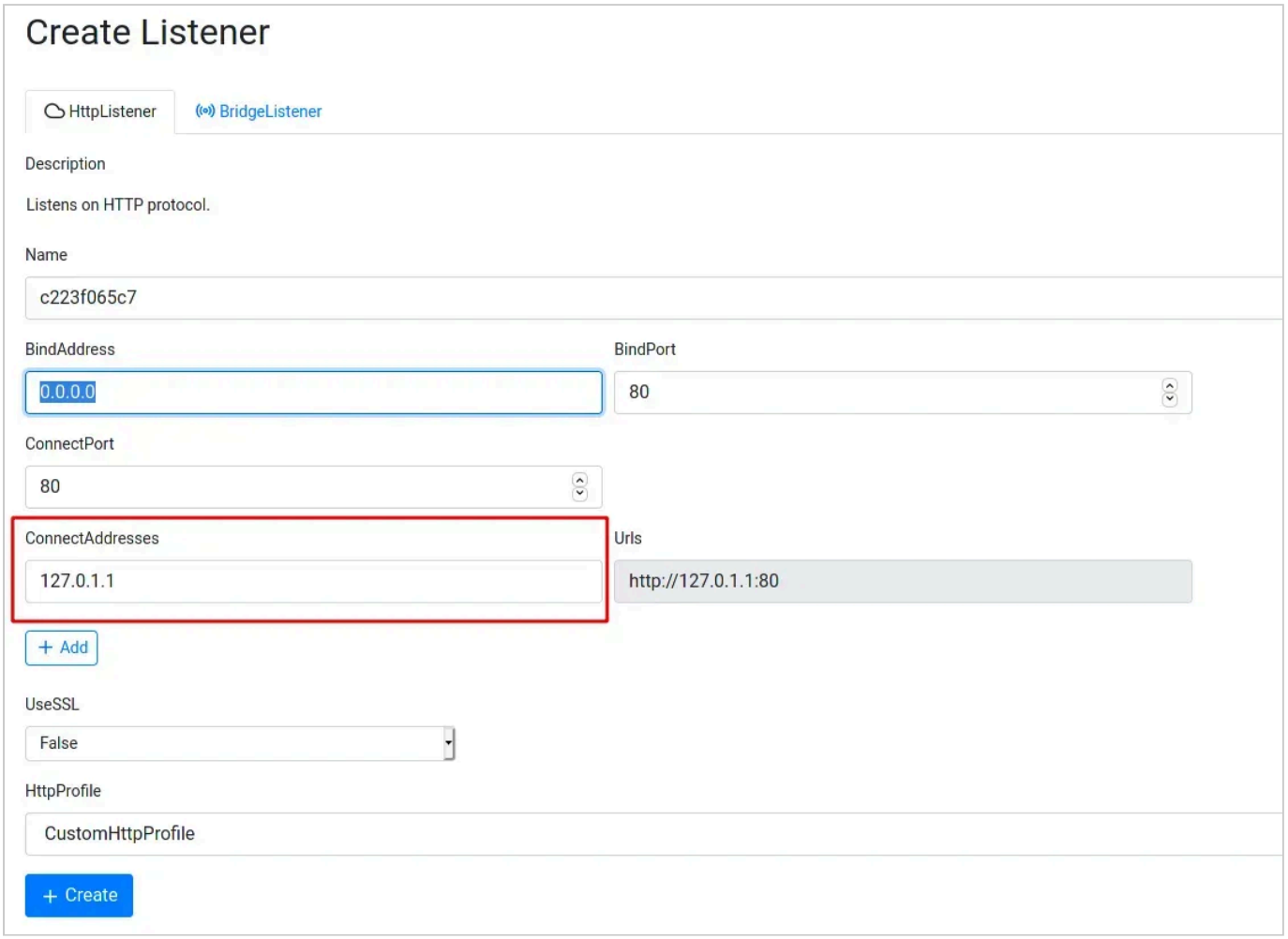
Register Initial User



Covenant Dashboard

Creating Listener

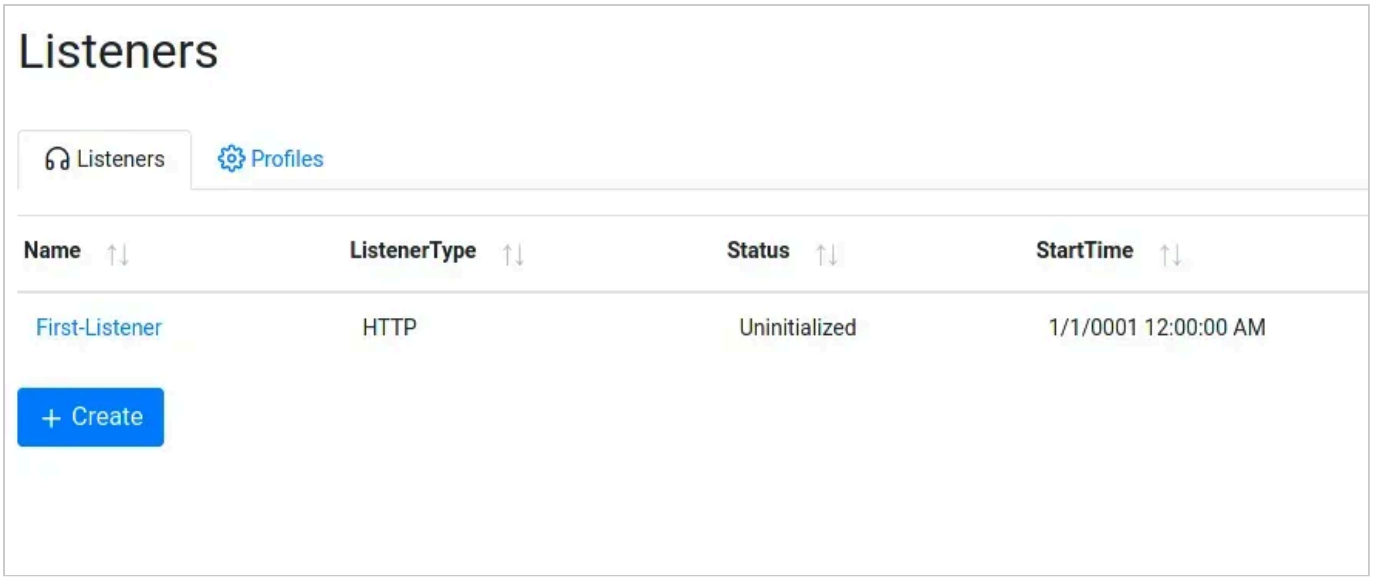
Before we can use the Covenant for red teaming activity, the first thing we need to setup is Listener. Basically covenant Listener is same as the usual listener we have used like netcat or meterpreter. In covenant, stager is called as Grunt. We will talk about it in the next section below. First of all, create the listener at listener menu.



Creating First Listener

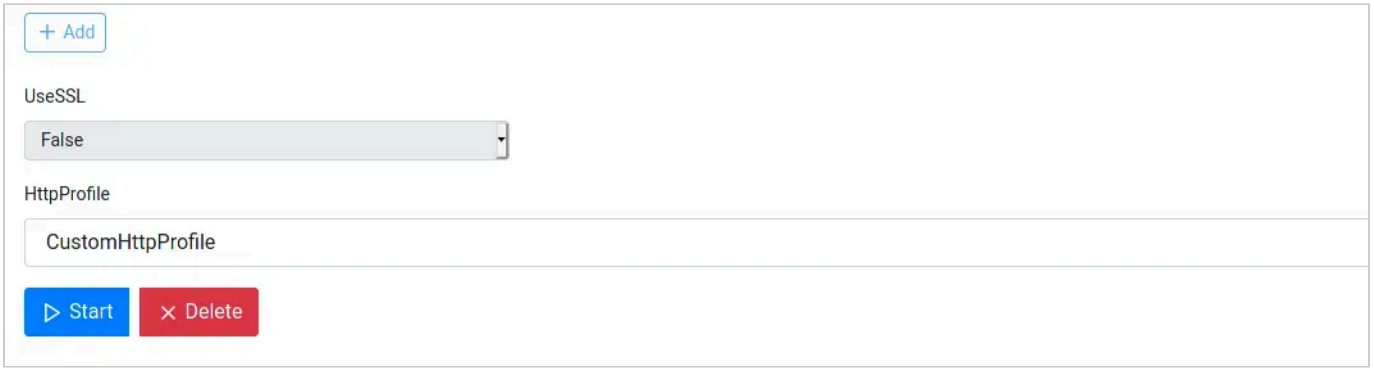
- **Name:** Identifier name for the listener, default is generated value but you can set it as you like
- **BindAddress:** The ip address listener will bind to
- **BindPort:** The port listener will bind to
- **ConnectAddress & ConnectPort:** Address and port that will be used as connect back for the stager.
- **HttpProfile:** You can leave it default

After we create the listener, it will show in listeners list. For example, I create listener with name First-Listener and type is HTTP.

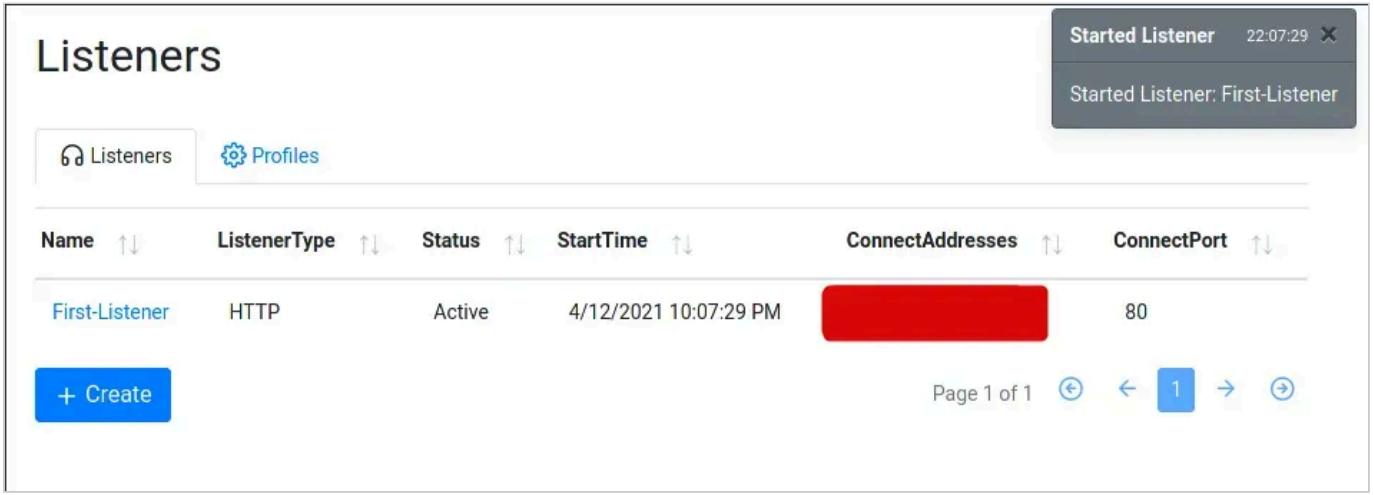


Listener

Start the listener we've created before and it will show you the new information in listener like the figure below



Start Listener



Listener Active

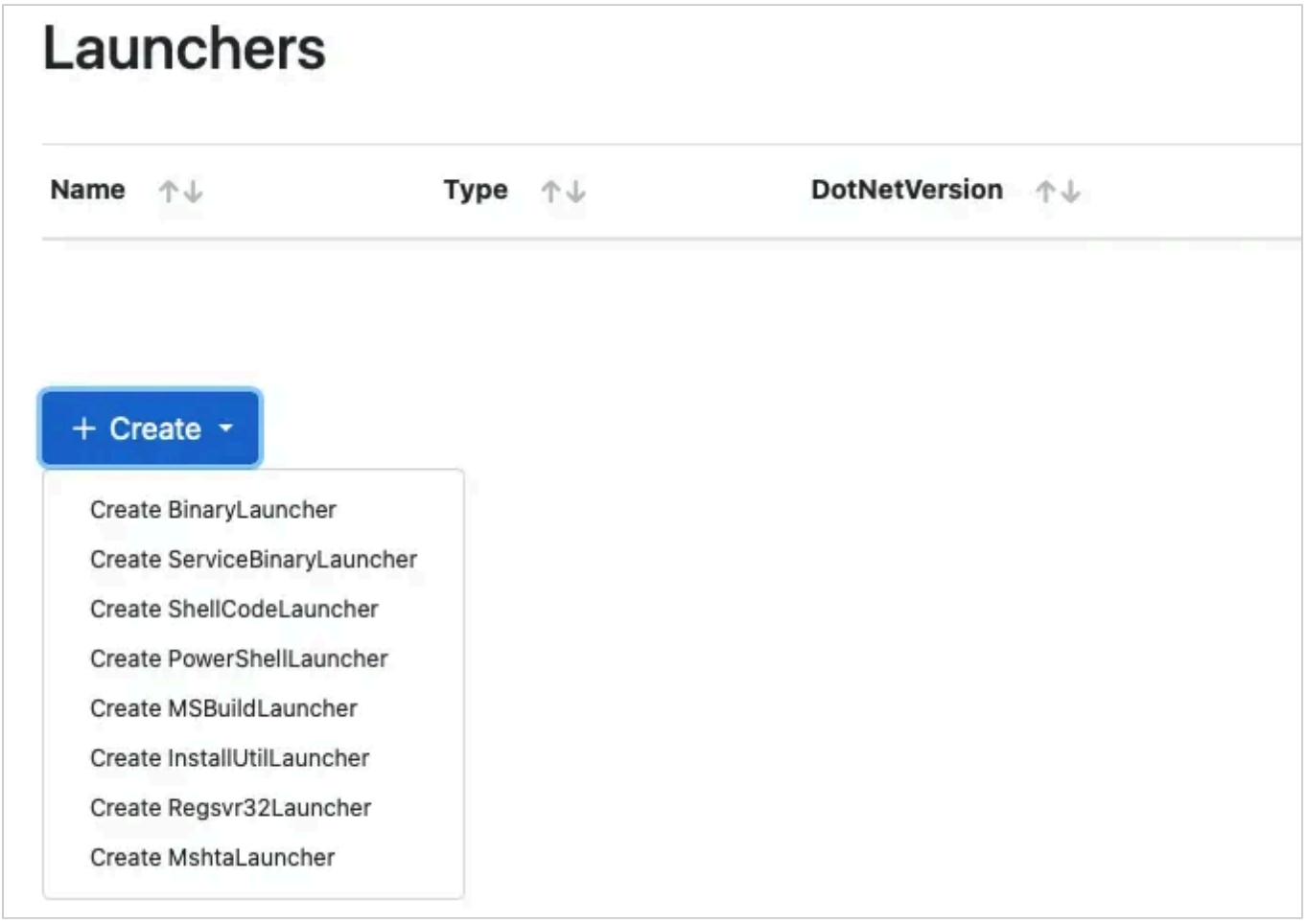
Licensed under CC BY-SA 4.0

We can double-check to make sure if listener is successfully active using this command in terminal:

```
└─$ sudo netstat -tulpn | grep 'LISTEN.*Covenant'
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:7443            0.0.0.0:*               LISTEN
```

Launcher

Launchers are all in one payload delivery feature that generate, host, and download binaries/script to launch new Grunts



Covenant Launcher

Here is explanation from Covenant wiki:

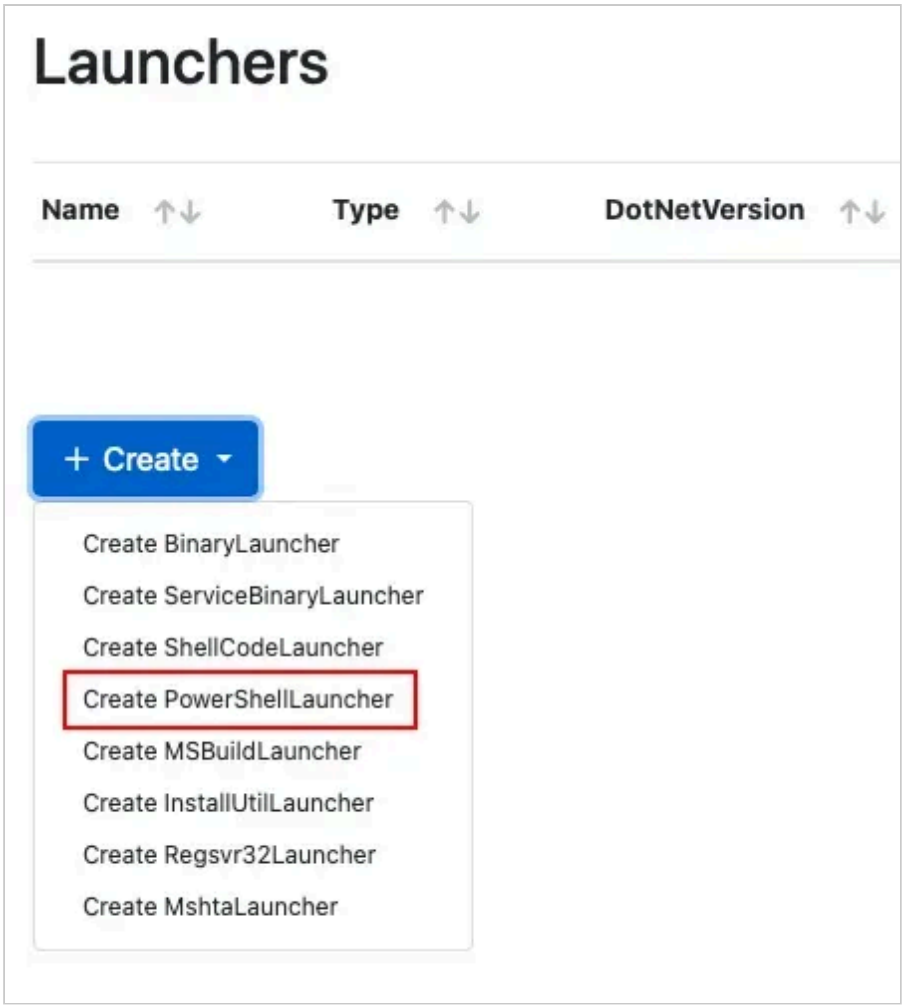
- Binary — The `Binary` launcher is used to generate custom binaries that launch a Grunt. This is currently the only launcher that does not rely on a system binary.
- ShellCode — The `ShellCode` launcher converts a Grunt binary to ShellCode using Donut.

Licensed under CC BY-SA 4.0

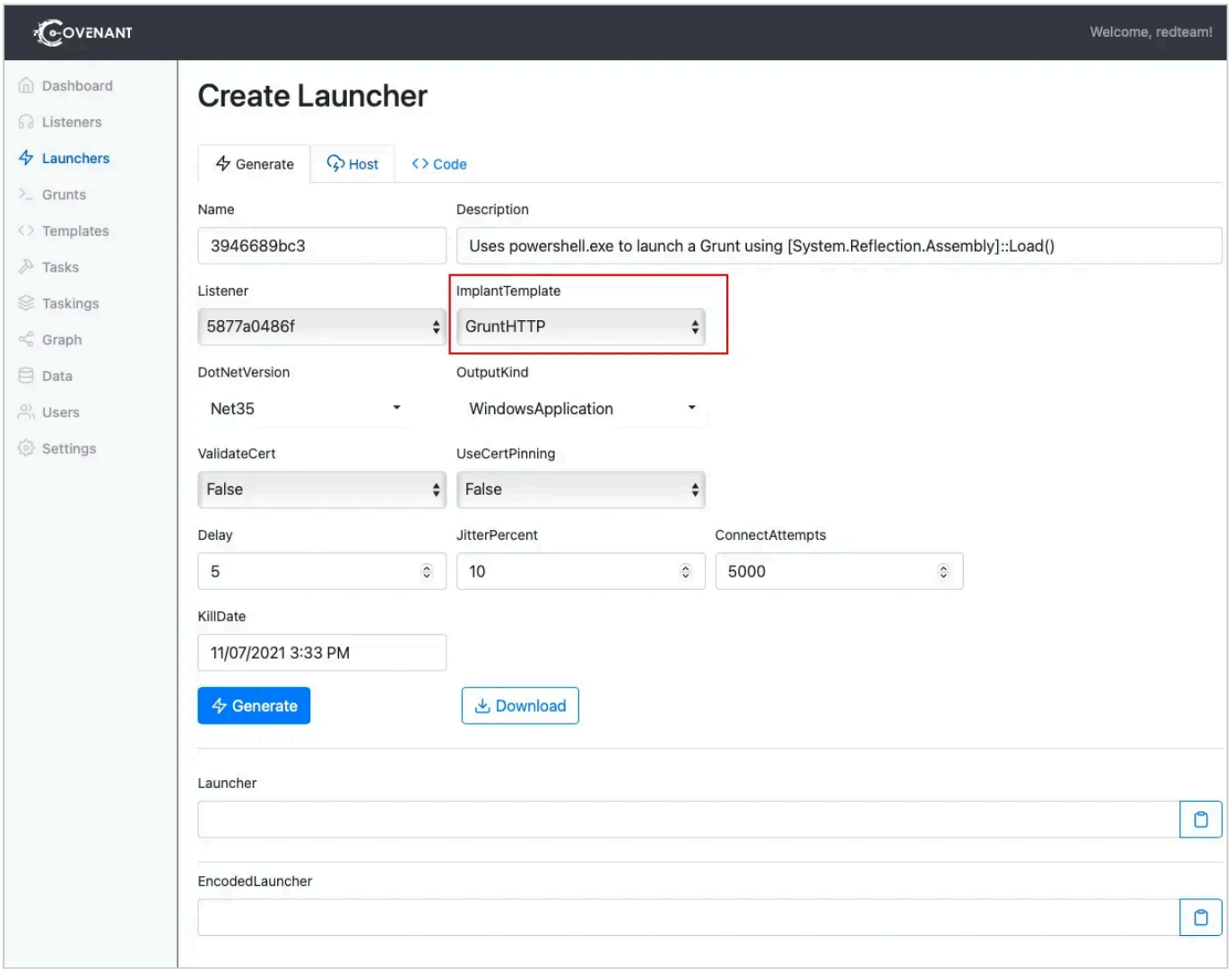
- PowerShell — The `PowerShell` launcher is used to generate PowerShell code and/or a PowerShell one-liner that launches a Grunt using `powershell.exe`.
- MSBuild — The `MSBuild` launcher is used to generate an MSBuild XML file that launches a Grunt using `msbuild.exe`.
- InstallUtil — The `InstallUtil` launcher is used to generate an InstallUtil XML file that launches a Grunt using `installutil.exe`.
- Mshta — The `Mshta` launcher is used to generate an HTA file and/or a mshta one-liner that launches a Grunt using `mshta.exe` that relies on DotNetToJScript.
- Regsvr32 — The `Regsvr32` launcher is used to generate an SCT file and/or regsvr32 one-liner that launches a Grunt using `regsvr32.exe` that relies on DotNetToJScript.
- Wmic — The `Wmic` launcher is used to generate an xsl file and/or wmic one-liner that launches a Grunt using `wmic.exe` that relies on DotNetToJScript.
- Cscript — The `Cscript` launcher is used to generate a JScript file a Grunt using `cscript.exe` that relies on DotNetToJScript.
- Wscript — The `Wscript` launcher is used to generate a JScript file a Grunt using `wscript.exe` that relies on DotNetToJScript.

For this article, I will use Powershell Launcher as the example. Don't forget to disable windows defender or bypass the AMSI first(it will be discuss later).

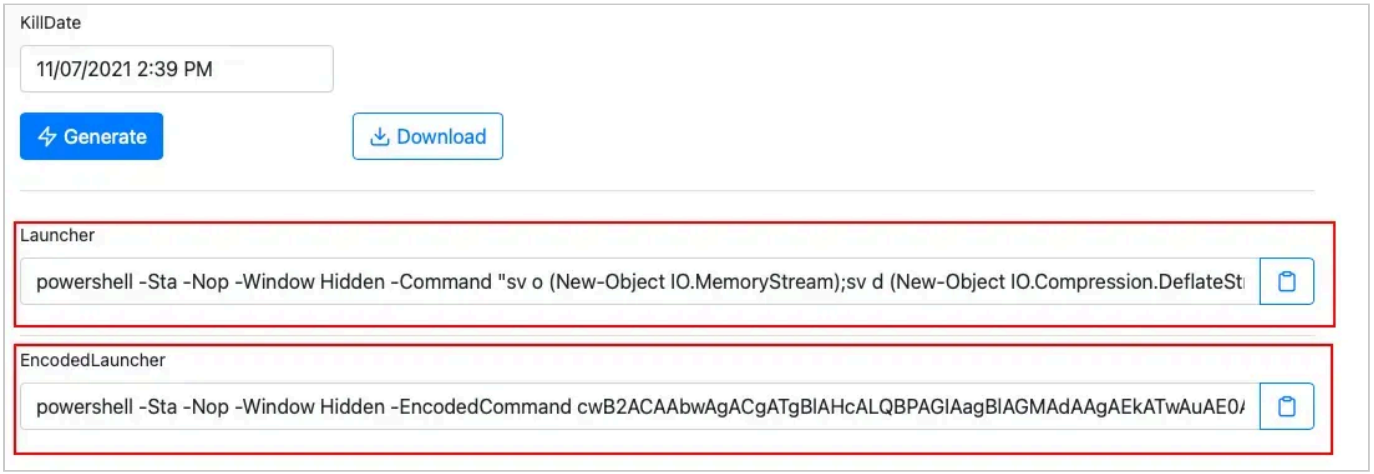
Creating Launcher



Create PowerShellLauncher



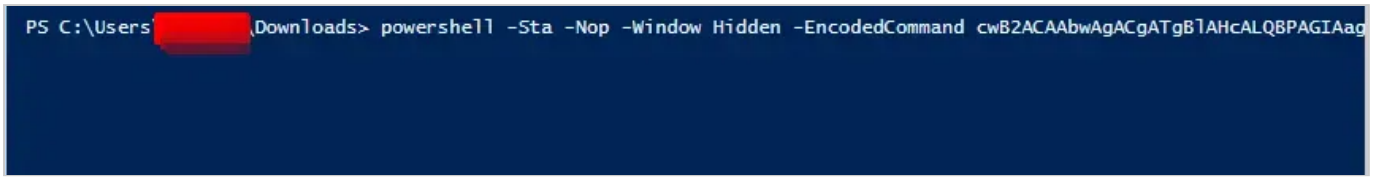
Choose and customize with your own environment. I suggest that you choose GruntHTTP for the implant template. After that, click generate and it will generate a launcher and encoded launcher for our need.



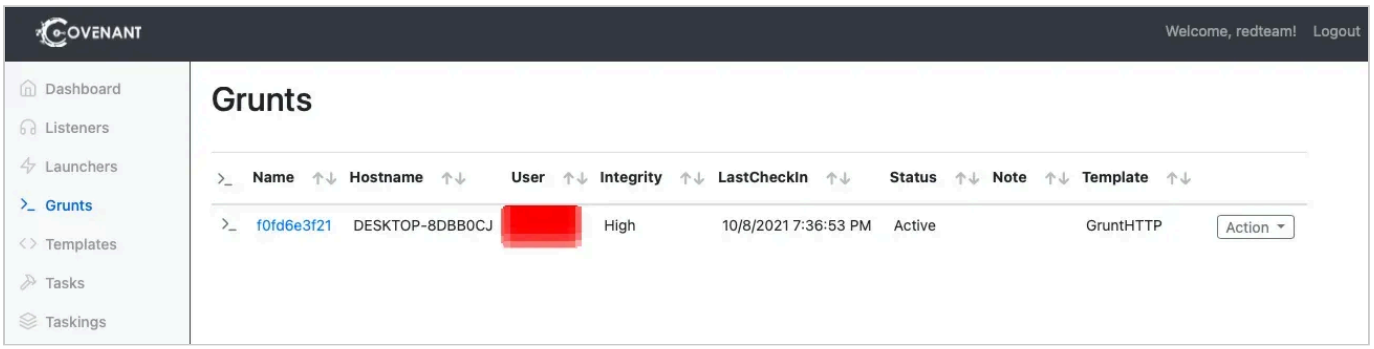
Generated launcher

Testing Launcher

We need to test our launcher in our windows box to see how the payload and communication work. Open command prompt and paste the generated launcher.



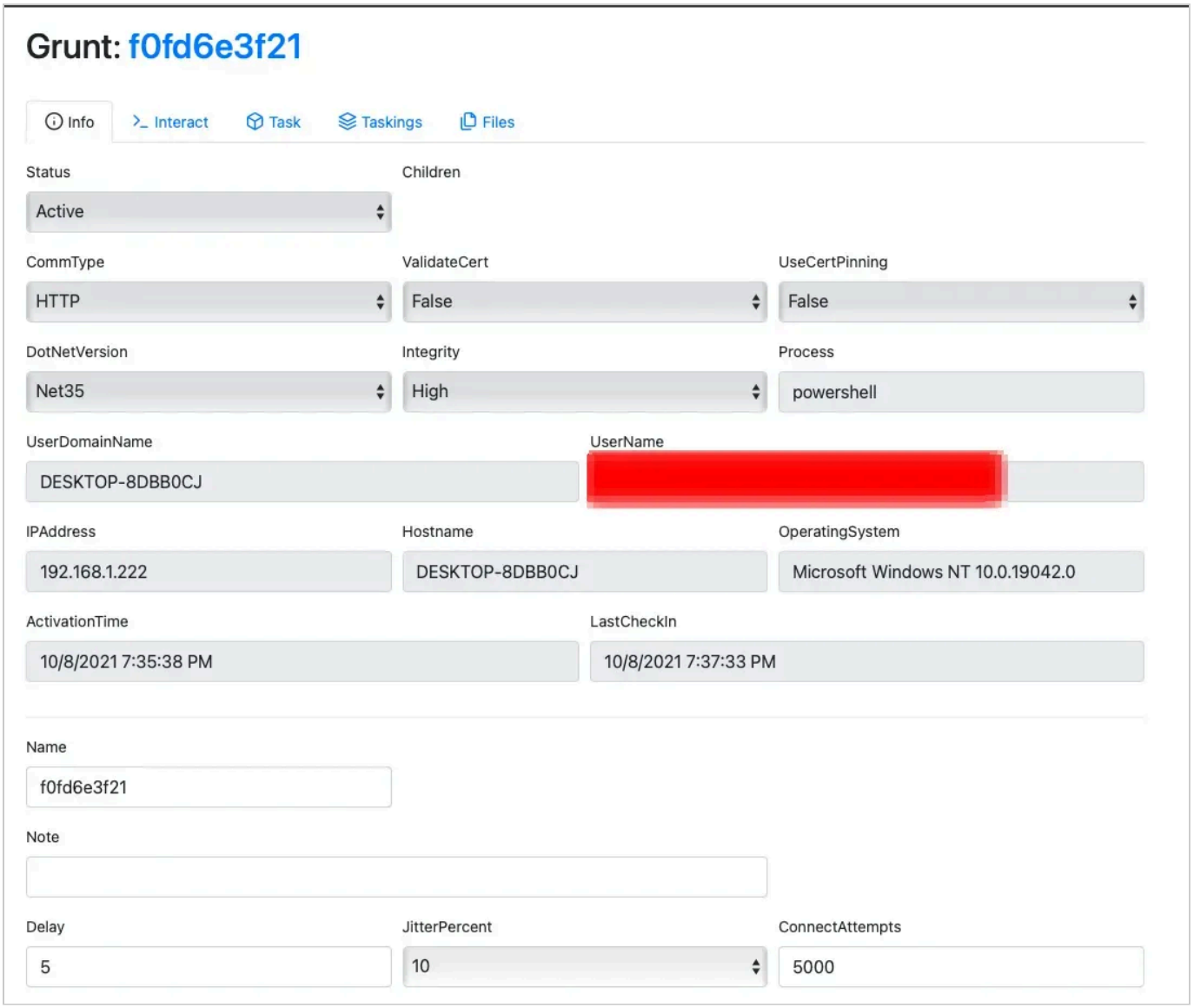
Powershell launcher



Grunt new connection

As we can see from above picture, our launcher is successfully connect to listener in grunt. This is what look like in grunts

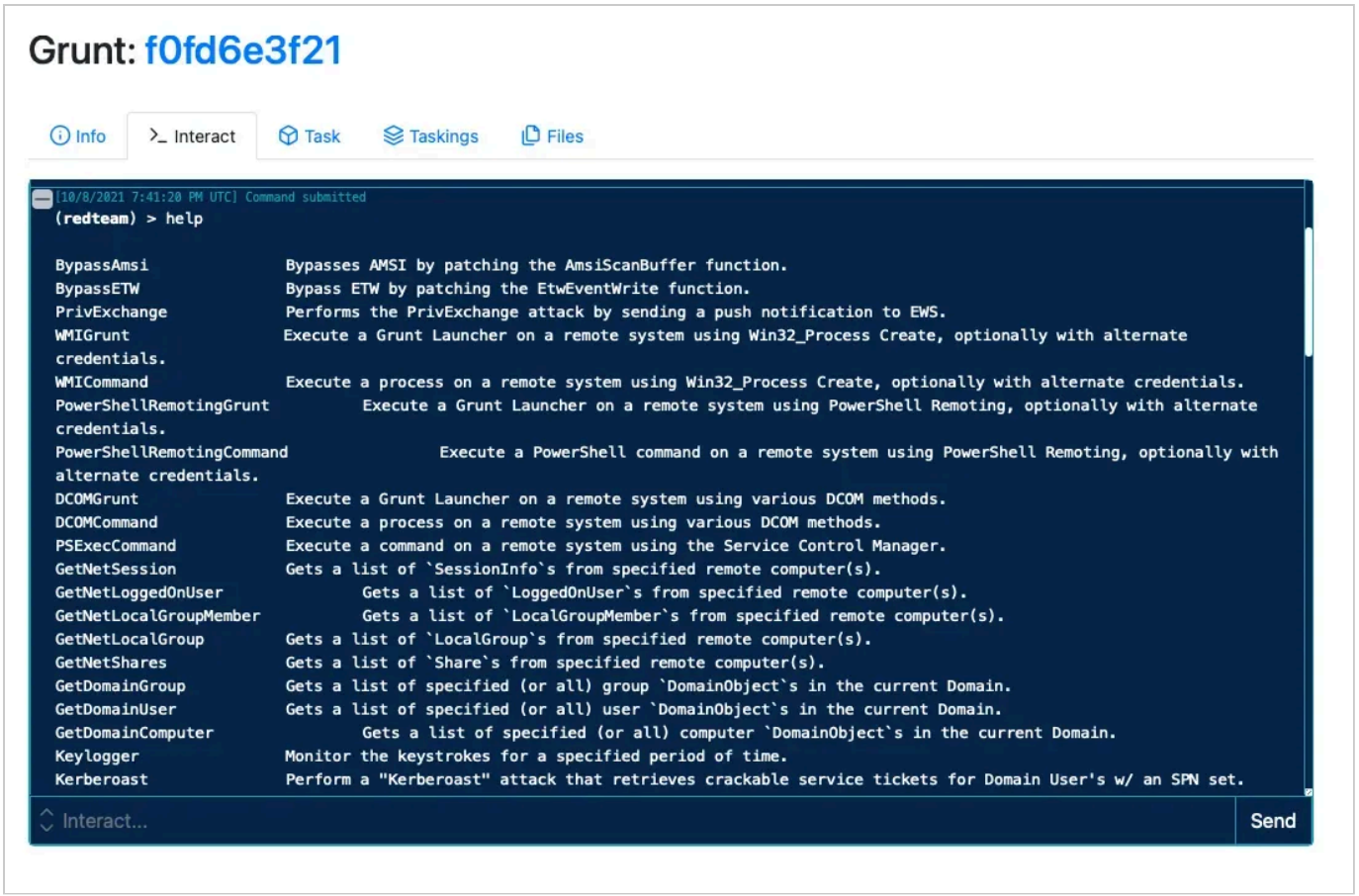
Licensed under CC BY-SA 4.0



Grunt information

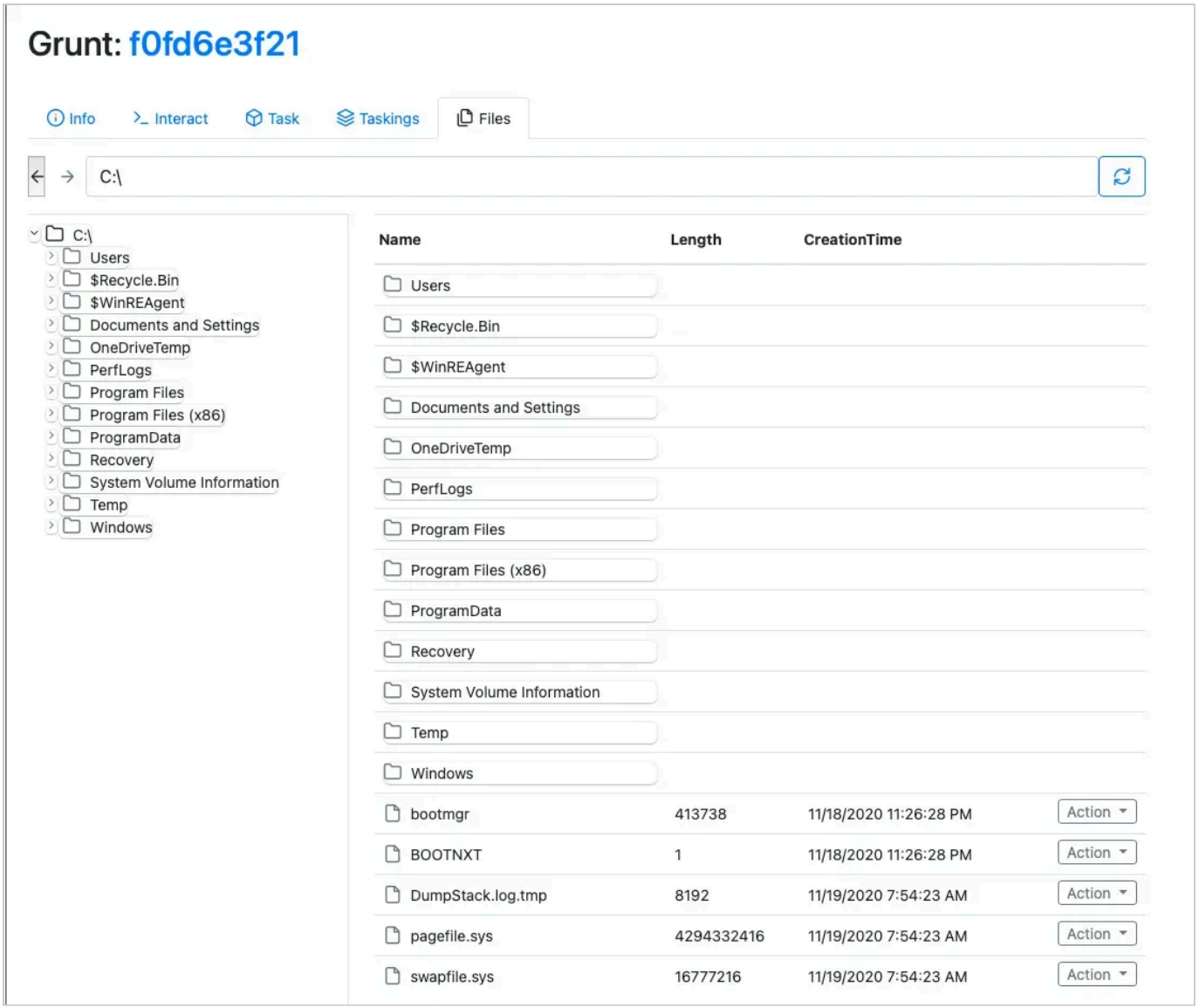
We can interact with the victim machine using covenant interact feature in grunt

Licensed under CC BY-SA 4.0



Grunt interaction

Covenant also support GUI file browser



Grunt file browser

I think that's all for the getting started article about Covenant C2. Will talk about the more covenant feature in the next article.

If you like this article, please share it and feedback are always welcome.

Reference:

- https://posts.specterops.io/entering-a-covenant-net-command-and-control-e11038bcf462 (https://posts.specterops.io/entering-a-covenant-net-command-and-control-e11038bcf462)

Originally published on Medium (https://medium.com/@petruknisme/getting-started-with-covenant-c2-for-red-teaming-8eeb94273b52).

Licensed under CC BY-SA 4.0

Red Teaming # Security # Tutorial

Licensed under CC BY-SA 4.0