

# Havoc C2 Framework Part 1: Installation (2024)

 [medium.com/@r1ckyr3c0n/havoc-c2-framework-part-1-installation-2024-2fbb8a1fe31c](https://medium.com/@r1ckyr3c0n/havoc-c2-framework-part-1-installation-2024-2fbb8a1fe31c)

r1ckyr3c0n

January 5, 2024



r1ckyr3c0n



Havoc Framework

After using Cobalt Strike in the Red Team Ops (RTO) course, I wanted to see what open-source Command and Control (C2) frameworks were available, so I could learn something new and continue to use a red team framework. A Google search led me to the [Havoc Framework](#) GitHub repository and upon initial glance, I thought Havoc resembled Cobalt Strike, so I decided to install it and play around with it in my virtual home lab. While it has similarities to Cobalt Strike (which is great if you are familiar with

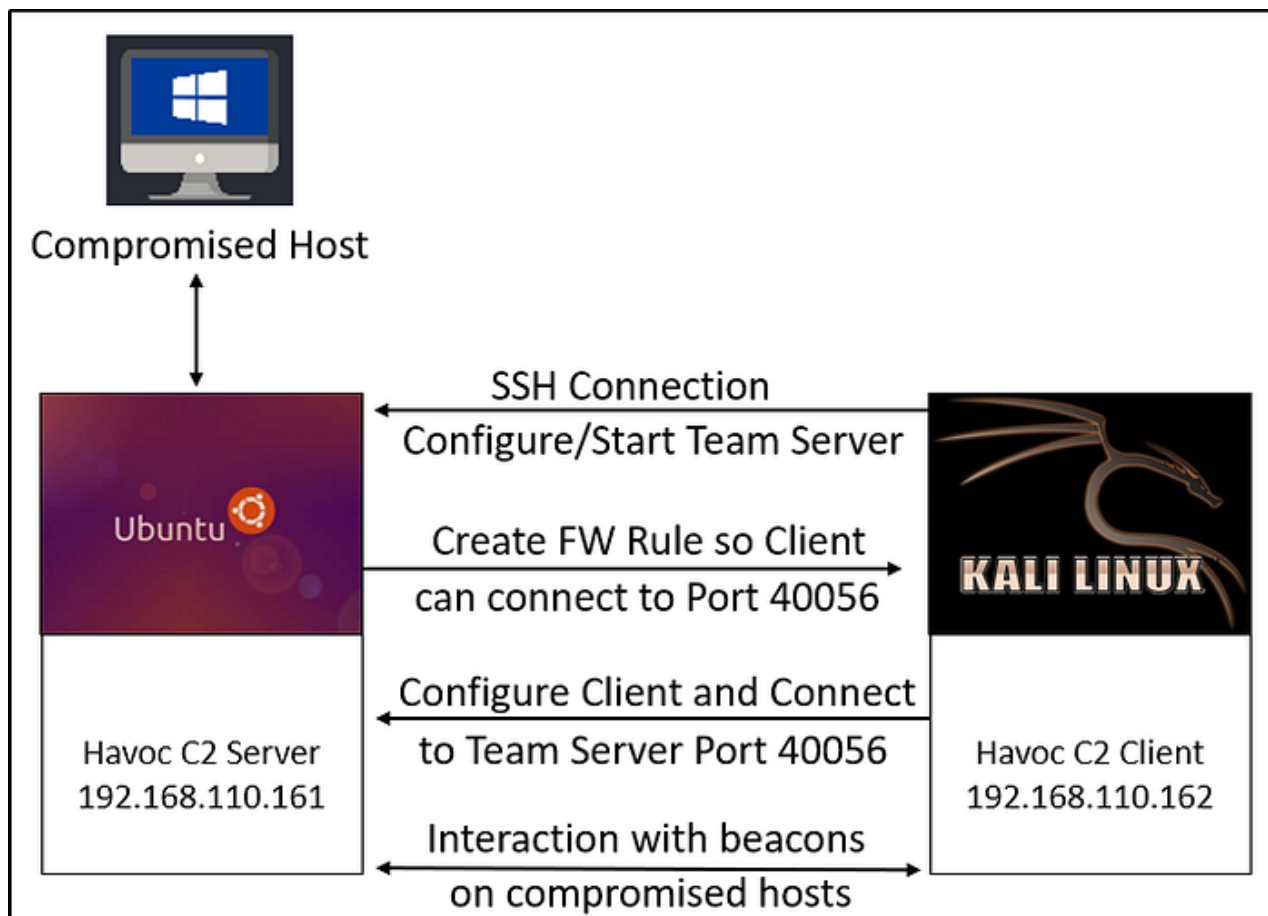
Cobalt Strike), I do not intend to make this a comparison article, rather a collection of my experience gained from installing it locally and using it within my local Active Directory (AD) lab environment.

## Havoc Overview

According to the Havoc GitHub repository, “Havoc is a modern and malleable post-exploitation command and control framework” and was created by [@C5pider](#). It’s latest documentation says it works well on Debian 10/11, Ubuntu 20.04/22.04, and Kali Linux and the installation documentation is superb and can be found on the [Havoc Framework](#) website. Havoc C2 consists of a Teamserver and Client that uses a profile that can be customized to emulate an adversary. The Havoc Teamserver runs that configured profile and interacts with “**sessions**” (think Cobalt Strike beacons) checking in from compromised hosts. The Havoc Client connects to the Teamserver and provides a User Interface (UI) that is like Cobalt Strike, and provides management of beacons, listeners, payloads, and so much more.

## Installing Havoc

For this installation, I chose to use Ubuntu Server 22.04.3 LTS to run a Havoc Teamserver, and Kali Linux 2023.4 to run the Havoc Client, with Vmware Workstation 15 Pro. You can setup the team server and client on the same host if you would like, but I wanted to create it a little like how red team infrastructure should be, i.e., putting some distance between the client and target (down the road, I will add some redirectors to proxy traffic and add another Teamserver).



My Havoc C2 Infrastructure

Like I said earlier, the installation documentation on the Havoc website is phenomenal and is what I used to perform the installation and was able to easily follow along. Since I was using a new Ubuntu server, there were only a few things outside of the installation instructions that I needed to do, which are highlighted below.

### Havoc Teamserver

After downloading the Ubuntu ISO file and Kali Vmware image, I created and configured the virtual machines and simply used Secure Shell (SSH) to remotely access the Ubuntu server from the Kali Linux machine. After that it was nothing more than running through the documented instructions on the [Havoc Framework](#) website.

```

r1cky@havocserver:~$ git clone https://github.com/HavocFramework/Havoc.git
Cloning into 'Havoc' ...
remote: Enumerating objects: 10835, done.
remote: Counting objects: 100% (3049/3049), done.
remote: Compressing objects: 100% (1026/1026), done.
remote: Total 10835 (delta 2175), reused 2672 (delta 1977), pack-reused 7786
Receiving objects: 100% (10835/10835), 33.76 MiB | 19.00 MiB/s, done.
Resolving deltas: 100% (7233/7233), done.
r1cky@havocserver:~$ cd Havoc/
r1cky@havocserver:~/Havoc$ sudo add-apt-repository ppa:deadsnakes/ppa
[sudo] password for r1cky:
Sorry, try again.
[sudo] password for r1cky:
Repository: 'deb https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu/ jammy main'
Description:
This PPA contains more recent Python versions packaged for Ubuntu.

Disclaimer: there's no guarantee of timely updates in case of security problems or other issues.
you do so at your own risk.

Update Note
=====
Please use this repository instead of ppa:fknull/deadsnakes.

[...snipped to save space...just know there will be a decent amount of output displayed...]

Hit:4 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:5 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease [18.1 kB]
Get:6 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy/main amd64 Packages [24.1 kB]
Get:7 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy/main Translation-en [4,800 B]
Fetched 46.9 kB in 1s (50.3 kB/s)
Reading package lists... Done
r1cky@havocserver:~/Havoc$

```

SSH connection from Kali Linux box to Ubuntu Server and started Havoc C2 installation process

```

r1cky@havocserver:~/Havoc$ sudo apt update
Hit:1 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
47 packages can be upgraded. Run 'apt list --upgradable' to see them.
r1cky@havocserver:~/Havoc$ sudo apt install python3.10 python3.10-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3.10 is already the newest version (3.10.12-1~22.04.3).
python3.10 set to manually installed.
The following additional packages will be installed:

[...snipped to save space...just know there will be a decent amount of output displayed...]

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
r1cky@havocserver:~/Havoc$

```

Installation process continued

The Ubuntu server did not have “Go Programming Language” installed, so before I could install the additional “Go” dependencies, I needed to install “Go Programming Language”.

```
ricky@havocserver:~$ sudo apt install golang-go
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bzip2 cpp cpp-11 g++ g++-11 gcc gcc-11 gcc-11-base golang-1.18-go golang-1.18-src golang-src libb
  libitm1 liblsan0 libmpc3 libquadmath0 libstdc++-11-dev libtsan0 libubsan1 pkg-config
Suggested packages:
  bzip2-doc cpp-doc gcc-11-locales g++-multilib g++-11-multilib gcc-11-doc gcc-multilib make auto
  debian-keyring bzip libstdc++-11-doc dpkg-dev
The following NEW packages will be installed:
  bzip2 cpp cpp-11 g++ g++-11 gcc gcc-11 gcc-11-base golang-1.18-go golang-1.18-src golang-go golang
  libisl23 libitm1 liblsan0 libmpc3 libquadmath0 libstdc++-11-dev libtsan0 libubsan1 pkg-config
0 upgraded, 28 newly installed, 0 to remove and 47 not upgraded.
Need to get 137 MB of archives.
After this operation, 608 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 bzip2 amd64 1.0.8-5build1 [34.8 kB]
[...]snipped to save space...just know there will be a decent amount of output displayed...]

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ricky@havocserver:~$ cd Havoc/teamserver/
ricky@havocserver:~/Havoc/teamserver$ go mod download golang.org/x/sys
ricky@havocserver:~/Havoc/teamserver$ go mod download github.com/ugorji/go
ricky@havocserver:~/Havoc/teamserver$ cd ..
ricky@havocserver:~/Havoc$
```

Installed “” on Ubuntu Server and continued installation process



```

r1cky@havocserver:~$ sudo apt install make
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  make-doc
The following NEW packages will be installed:
  make
0 upgraded, 1 newly installed, 0 to remove and 47 not upgraded.

[...snipped to save space...just know there will be a decent amount of output displayed...]

r1cky@havocserver:~$ cd Havoc/
r1cky@havocserver:~/Havoc$ make ts-build
[*] building teamserver
go: downloading github.com/spf13/cobra v1.2.1
go: downloading github.com/fatih/color v1.12.0
go: downloading github.com/fatih/structs v1.1.0
go: downloading github.com/gin-gonic/gin v1.7.7
go: downloading github.com/gorilla/websocket v1.5.0
go: downloading golang.org/x/crypto v0.0.0-20220314234659-1baeb1ce4c0b

[...snipped to save space...just know there will be a decent amount of output displayed...]

r1cky@havocserver:~/Havoc$ cd profiles/
r1cky@havocserver:~/Havoc/profiles$ ls
havoc.yaotl http_smb.yaotl webhook_example.yaotl
r1cky@havocserver:~/Havoc/profiles$ sudo nano havoc.yaotl
r1cky@havocserver:~/Havoc/profiles$ █

```

Installed "" on Ubuntu Server and continued installation process

Since I was planning to run the Teamserver and Client on different hosts, I added the IP address of the Ubuntu server to the "**Host**" field and added different "**Operators**" that would be able to login to the Havoc Client.

```

GNU nano 6.2                                     havoc.yaotl *
Teamserver {
  Host = "192.168.110.161"
  Port = 40056

  Build {
    Compiler64 = "data/x86_64-w64-mingw32-cross/bin/x86_64-w64-mingw32-gcc"
    Compiler86 = "data/i686-w64-mingw32-cross/bin/i686-w64-mingw32-gcc"
    Nasm = "/usr/bin/nasm"
  }
}

Operators {
  user "ricky" {
    Password = "havoc123!@#"
  }

  user "z3r0c001" {
    Password = "havoc123!@#"
  }
}

# this is optional. if you dont use it you can remove it.
Service {
  Endpoint = "service-endpoint"
  Password = "service-password"
}

Demon {
  Sleep = 2
  Jitter = 15

  TrustXForwardedFor = false

  Injection {
    Spawn64 = "C:\\Windows\\System32\\notepad.exe"
    Spawn32 = "C:\\Windows\\SysWOW64\\notepad.exe"
  }
}

```


Edits made to Teamserver profile

After saving the **havoc.yaotl** profile, I was able to start the Teamserver and with the verbose and debug flags set, was able to see that it appeared to start and was running properly.

```

ricky@havocserver:~/Havoc/profiles$ cd ..
ricky@havocserver:~/Havoc$ ./havoc server --profile ./profiles/havoc.yaotl -v --debug

```



```

    pwn and elevate until it's done

[20:12:42] [DEBUG] [cmd.glob..func2:59]: Debug mode enabled
[20:12:42] [INFO] Havoc Framework [Version: 0.7] [CodeName: Bites The Dust]
[20:12:42] [INFO] Havoc profile: ./profiles/havoc.yaotl
[20:12:42] [INFO] Build:
- Compiler x64 : data/x86_64-w64-mingw32-cross/bin/x86_64-w64-mingw32-gcc
- Compiler x86 : data/i686-w64-mingw32-cross/bin/i686-w64-mingw32-gcc
- Nasm         : /usr/bin/nasm
[20:12:42] [INFO] Time: 02/01/2024 20:12:42
[20:12:42] [INFO] Teamserver logs saved under: data/loot/2024.01.02_20:12:42
[20:12:42] [DEBUG] [server.(*Teamserver).Start:53]: Starting teamserver...
[20:12:42] [INFO] Starting Teamserver on wss://192.168.110.161:40056
[20:12:42] [INFO] [SERVICE] starting service handle on wss://192.168.110.161:40056/service-endpoint
[20:12:42] [INFO] Opens existing database: data/teamserver.db
[20:12:42] [DEBUG] [server.(*Teamserver).Start:492]: Wait til the server shutdown
[20:12:42] [DEBUG] [certs.HTTPSGenerateRSACertificate:301]: Generating TLS certificate (RSA) for '192.168.110.161' ...
[20:12:42] [DEBUG] [certs.generateCertificate:223]: Valid from 2023-11-10 20:12:42.818335961 +0000 UTC to 2026-11-09 20:12:42.818335961 +0000 UTC
[20:12:42] [DEBUG] [certs.generateCertificate:228]: Serial Number: 308966270517558249095220473446745096689
[20:12:42] [DEBUG] [certs.generateCertificate:234]: Authority certificate
[20:12:42] [DEBUG] [certs.generateCertificate:247]: ExtKeyUsage = [1 2]
[20:12:42] [DEBUG] [certs.generateCertificate:263]: Certificate authenticates IP address: 192.168.110.161
[20:12:42] [DEBUG] [certs.generateCertificate:278]: Certificate is an AUTHORITY

```

Havoc Teamserver started and running on Ubuntu Server

## Havoc Client

Since the Teamserver was up and running, it was now time to get the Havoc Client setup on the Kali Linux box. As a non-root user, working through the steps outlined in the Havoc Framework Installation documentation was going seemingly, but I seemed to run into issues installing the dependencies for **“Kali and other Debian based Distros only”** section. I then ran **sudo apt update** and tried that long install command again and it seemed to work. I then just decided to run through the installation process as the root user and had no issues going through any of the installation steps (I also did this as a non-root user in an included video at the end of this article).

```
(root@kali)-[~]
# git clone https://github.com/HavocFramework/Havoc.git
Cloning into 'Havoc' ...
remote: Enumerating objects: 10835, done.
remote: Counting objects: 100% (3122/3122), done.
remote: Compressing objects: 100% (1034/1034), done.
remote: Total 10835 (delta 2238), reused 2738 (delta 2042), pack-reused 7713
Receiving objects: 100% (10835/10835), 33.75 MiB | 25.96 MiB/s, done.
Resolving deltas: 100% (7246/7246), done.

(root@kali)-[~]
# cd Havoc

(root@kali)-[~/Havoc]
# sudo apt install -y git build-essential apt-utils cmake libfontconfig1 libglu1-mesa-dev
v libsqlite3-dev libbz2-dev mesa-common-dev qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tool
hon3-dev libboost-all-dev mingw-w64 nasm
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done

[...snipped to save space...just know there will be a decent amount of output displayed...]

(root@kali)-[~/Havoc]
# echo 'deb http://ftp.de.debian.org/debian bookworm main' >> /etc/apt/sources.list

(root@kali)-[~/Havoc]
# sudo apt update
Hit:2 http://ftp.de.debian.org/debian bookworm InRelease
Hit:1 http://mirrors.jevincanders.net/kali kali-rolling InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done

[...snipped to save space...just know there will be a decent amount of output displayed...]

(root@kali)-[~/Havoc]
# sudo apt install python3-dev python3.10-dev libpython3.10 libpython3.10-dev python3.10
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done

[...snipped to save space...just know there will be a decent amount of output displayed...]

(root@kali)-[~/Havoc]
#
```

Installation process for Havoc Client on Kali Linux box



```

(root@kali)~/Havoc
# cd teamserver

(root@kali)~/Havoc/teamserver
# go mod download golang.org/x/sys

(root@kali)~/Havoc/teamserver
# go mod download github.com/ugorji/go

(root@kali)~/Havoc/teamserver
# cd ..

(root@kali)~/Havoc
# make ts-build
[*] building teamserver
go: downloading github.com/spf13/cobra v1.2.1
go: downloading github.com/fatih/color v1.12.0
go: downloading github.com/fatih/structs v1.1.0
go: downloading github.com/gin-gonic/gin v1.7.7
go: downloading github.com/gorilla/websocket v1.5.0

[...snipped to save space...just know there will be a decent amount of output displayed...]

(root@kali)~/Havoc
# make client-build
[*] building client
Submodule 'client/external/json' (https://github.com/nlohmann/json) registered for path 'client/external/json'
Submodule 'client/external/spdlog' (https://github.com/gabime/spdlog) registered for path 'client/external/spdlog'
Submodule 'client/external/toml' (https://github.com/ToruNiina/toml11) registered for path 'client/external/toml'
Cloning into '/root/Havoc/client/external/json' ...
Cloning into '/root/Havoc/client/external/spdlog' ...
Cloning into '/root/Havoc/client/external/toml' ...
Submodule path 'client/external/json': checked out '6eab7a2b187b10b2494e39c1961750bfd1bda500'

[...snipped to save space...just know there will be a decent amount of output displayed...]

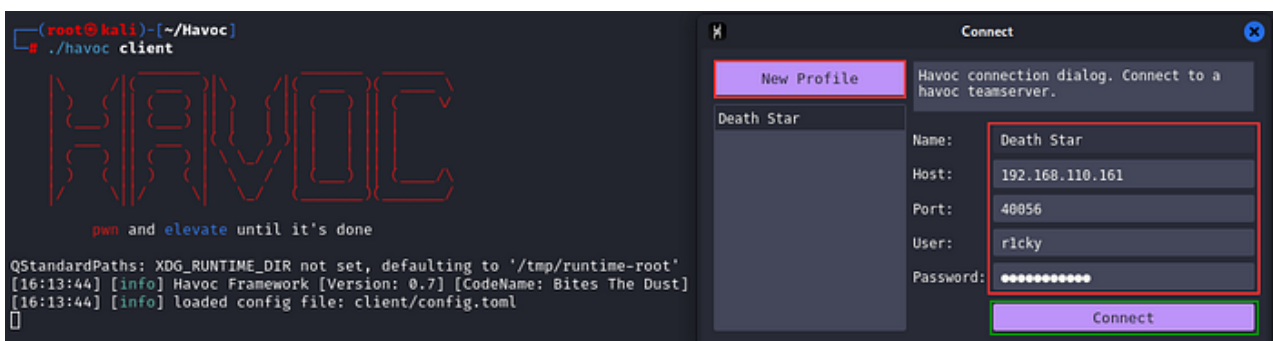
[ 96%] Building CXX object CMakeFiles/Havoc.dir/src/Util/Base.cpp.o
[ 98%] Building CXX object CMakeFiles/Havoc.dir/Havoc_autogen/QYFM2Z2WYQ/qrc_Havoc.cpp.o
[100%] Linking CXX executable /root/Havoc/client/Havoc
gmake[3]: Leaving directory '/root/Havoc/client/Build'
[100%] Built target Havoc
gmake[2]: Leaving directory '/root/Havoc/client/Build'
gmake[1]: Leaving directory '/root/Havoc/client/Build'

(root@kali)~/Havoc

```

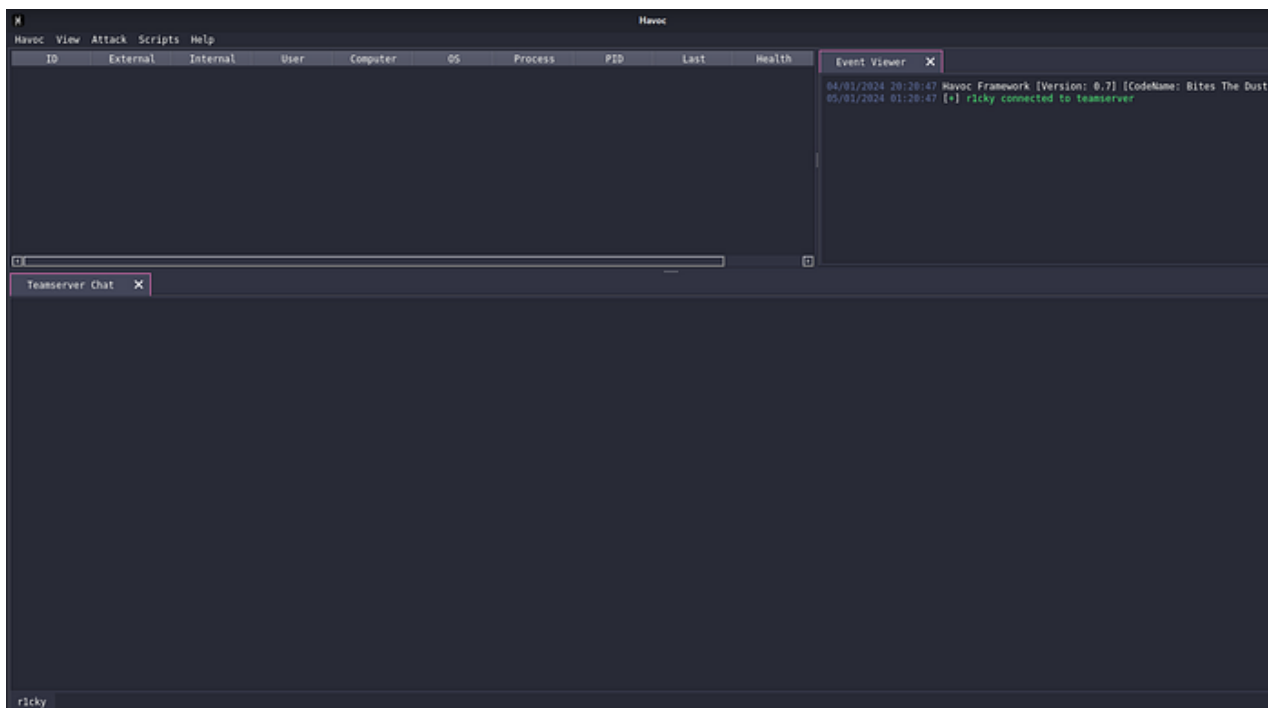
Continued installation process for Havoc Client on Kali Linux box

After successful installation, I was able to start the Havoc Client which launched a pop-up window for creating a “**New Profile**” or using loading a preexisting profile. I clicked on “**New Profile**”, left the **Name** and **Port**, as their default settings, and updated the **Host**, **User**, and **Password** to what I configured earlier in the **havoc.yaotl** profile. I then clicked on “**Connect**” to launch the Havoc Client UI.



Started Havoc Client and created a new profile

When the Havoc Client launches, you should have a User Interface that looks like this one and if you have worked with Cobalt Strike before, you will likely have no problem navigating it, as the layout and functionality is similar.



Havoc Client User Interface

## Conclusion

As you can see, the documented installation steps on the Havoc Framework website provide a relatively seamless process for getting the framework up and running. The next article will dive into the Havoc UI, to include creating listeners, payloads, establishing a session on a compromised host, post-exploitation, and everything else the UI has to offer. If you would like to watch this installation in real-time, I have uploaded an edited install of Havoc Framework to YouTube.