---

title: "SCADAsploit Module: Modbus Payload Transfer"

dates: 2022-04-22T15:09:35+02:00

draft: false

toc: false

images:

tags:

- SCADAsploit

---

The second module we are going to explore is called "\*\*Modbus Payload Transfer\*\*" and belongs to the auxiliary category.

## What is it for

The "\*\*Modbus Payload Transfer\*\*" module is a script that allows you to store and retrieve a payload/shellcode in the "holding registers" of a PLC via the Modbus TCP protocol. There are many PLCs exposed to the Internet, with this module it is possible to exploit their memory to upload a payload (shellcode/malware) so that it can be recovered at a later time.

A usage scenario could include the following steps:

1. An attacker locates an Internet-exposed PLC with enough space to store a particular payload.

2. The attacker loads the payload into the PLC's memory.

3. The attacker infects a host capable of communicating in Modbus TCP by retrieving the payload from the PLC and then executing it.

The same thing could obviously also happen with a PLC not exposed on the Internet but present in a local network. The main advantages of this method are:

1. The use of third party PLCs provides anonymity and makes traceability difficult. It is not necessary to upload the payload to a server.

2. The payload is stored in PLC memory, making forensic analysis difficult. Furthermore, once the payload has been retrieved, its content could easily be overwritten (even by the host itself).

This approach may also be useful in some ICS environments where protocols other than Modbus may alert IDS/IPS systems. In such a context you just need a Modbus device or even simply an emulator that can act as a server when the target host connects to it. Then there are PLCs directly exposed on the Internet and managed remotely which can become a good place to deposit a payload/malware.

\*\*Important Note\*\*: Do not perform any of these actions on third party PLCs. Any write to PLC registers can interrupt the process control strategy for which it has been programmed.

## How does it work

First we can search for the interested module using the `search modbus` command, which will present the list of auxiliary modules and the exploits that make use of this protocol. Once we have identified what we are looking for, we select it with the command `use auxiliary/modbus/modbus\_payload\_transfer`. The `show info` command gives us some details on how the module works.

![](</uploads/2021/scadasploit\_payload\_1.png>)

Now let's move on to configuring the module. The `show options` command displays the list of parameters, in this case we find:

\* ADDRESS: starting address of the holding register for the upload/download of the payload

\* FILENAME: our file containing the payload (a few Kb size is recommended)

\* MODE: to select whether to perform an upload or download from the PLC

\* NBYTES: the number of bytes that we have to download during a download (not necessary for an upload)

\* RHOST and RPORT to set the IP address and TCP port of our PLC

![](</uploads/2021/scadasploit\_payload\_2.png>)

When we have correctly configured the parameter list we just have to run our auxiliary module, using the `run` command. In this case we are uploading a file called `payload.bin` with a size of 300 bytes, on a PLC with address 10.43.10.58 starting from holding register 200.

![](</uploads/2021/scadasploit\_payload\_3.png>)

Due to the Modbus protocol characteristic, the file is divided into packets of 250 bytes. If the payload has an odd number of bytes a "0x90" character will be appended to avoid problems during retrieval.

Depending on the type of PLC in use we will have a greater or lesser amount of accessible memory available, therefore the script will first check that there is enough space for the payload. To verify the dimensions, a Modbus request is sent with an operation ID of 03 (Read Holding Register), attempting to read a certain record starting from the address that we have set as a parameter (each record is 16 bits). If you get a 0x83 exception it means that the PLC is unable to contain our payload and the operation fails.

> The payload recovery takes place using the same "\*\*Modbus Payload Transfer\*\*" module, simply by selecting the MODE = DOWN mode and the amount of bytes that we need to download via the NBYTES parameter.

>

> Also in this case the download takes place in blocks of 250 bytes (125 holding registers) at a time. The file is then reassembled and saved in its entirety.

## Beware of data loss

In addition to using the script to load a particular payload such as malware or shellcode, it goes without saying that it can also be used to load any type of small file. I think it's an interesting way to exfiltrate and share information. Who would suspect that the holding registers of a given PLC store a .docx or .zip file?

It is important to note that the holding registers into which the payload is loaded can be modified by the PLC itself. Since we don't know memory management and process control strategy it is likely that we need to look for a memory range that is not susceptible to change. The idea might be to upload the payload to a certain section of the logs and then check, for some time, that the payload has not undergone any modifications.

## \*\*Important note\*\*

This article is intended for educational and informational purposes only. \*\*Any unauthorized action\*\* towards any control system present on a public or private network \*\*is illegal\*\*! The information contained in this and other articles are intended to make people understand how necessary it is to improve defense systems, and not to provide tools for attacking them. Violating a computer system is punishable by law and can cause serious damage to property and people, especially when it comes to ICS. All the tests that are illustrated in the tutorials have been carried out in isolated, safe, or manufacturer-authorized laboratories.

Stay safe, stay free.