How to Identify XenoRAT C2 Servers

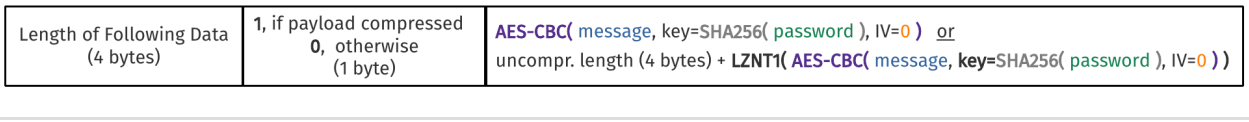
By Axel Mahr

Published: 2024-12-13 · Archived: 2026-04-05 18:38:44 UTC

XenoRAT is a relatively new RAT, that is open-source and used by low-sophisticated cyber criminals but also APT groups. In this post, we will look at how we can detect XenoRAT C2 servers through scanning. But, before we come to that, we need to take a quick look at XenoRAT's C2 protocol.

XenoRAT's C2 Packet Formats

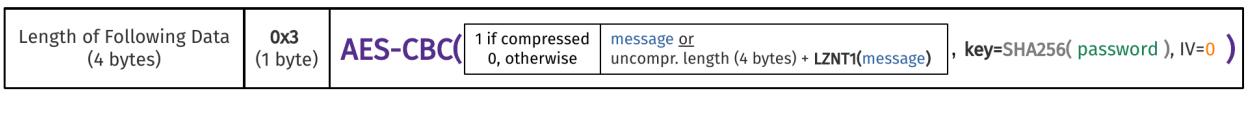
In general, implant and C2 server communicate over raw TCP while using dedicated packet formats which are set as TCP payload. The packet formats are different, depending on whether the internal server state `doProtocolUpgrade` is true or false. When a new client connects, this state is always false. Then, the following the packet format is used:



*TCP Payload Format if `doProtocolUprgade` is **false***

The first four bytes give the length of the following data. The next byte indicates, whether the payload is compressed or not. The rest of the packet constitutes the actual message, which is AES-encrypted. The AES key is the same for both communication directions and is derived by calculating the SHA256 hash of the password specified on the server side by the RAT operator. The default password is "1234" and the IV is always zero. The encrypted message is additionally compressed using LZNT1 and preceded by four bytes indicating the uncompressed length, if compression reduces the length of the encrypted message. However, this is very unlikely, as the encrypted message has already high entropy.

The packet format, that is used when `doProtocolUprgade` is true, is the following:

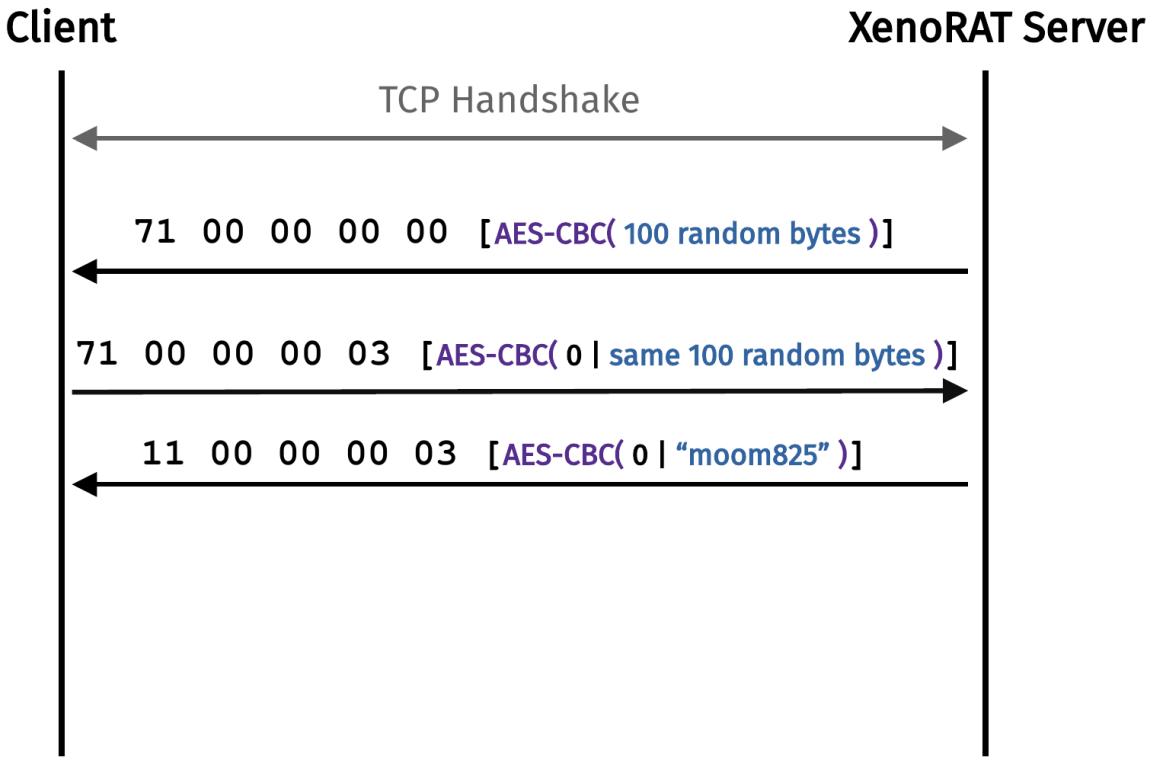


*TCP Payload Format if `doProtocolUprgade` is **true***

It looks pretty similar. However, notice that the fifth byte is always `0x3` and that the first byte of the plaintext to be encrypted is always 0 or 1 and precedes the actual message.

Detection

In order to detect XenoRAT C2 servers proactively through scanning, we can exploit, how the handshake between an implant and C2 server works:



The XenoRAT Handshake between a regular implant and the C2 server

Right after the TCP handshake, the C2 server sends a packet using the first format (`doProtocolUprgade` is false). 0x71 indicates the length of the following data, which is 0x70 bytes occupied by the AES-encrypted 100 random bytes and the preceding single zero indicating that the payload is not compressed. In the usual XenoRAT handshake, the implant answers with the respective random bytes while using the second packet format, which is in general the only format a regular implant uses. By setting the fifth byte to 0x3, the C2 server changes `doProtocolUpgrade` to true internally. By that, also the C2 server continues to solely use this format for future packets to send. When the implant has sent the 100 random bytes back correctly, the server sends a further packet consisting of the message `moom825`.

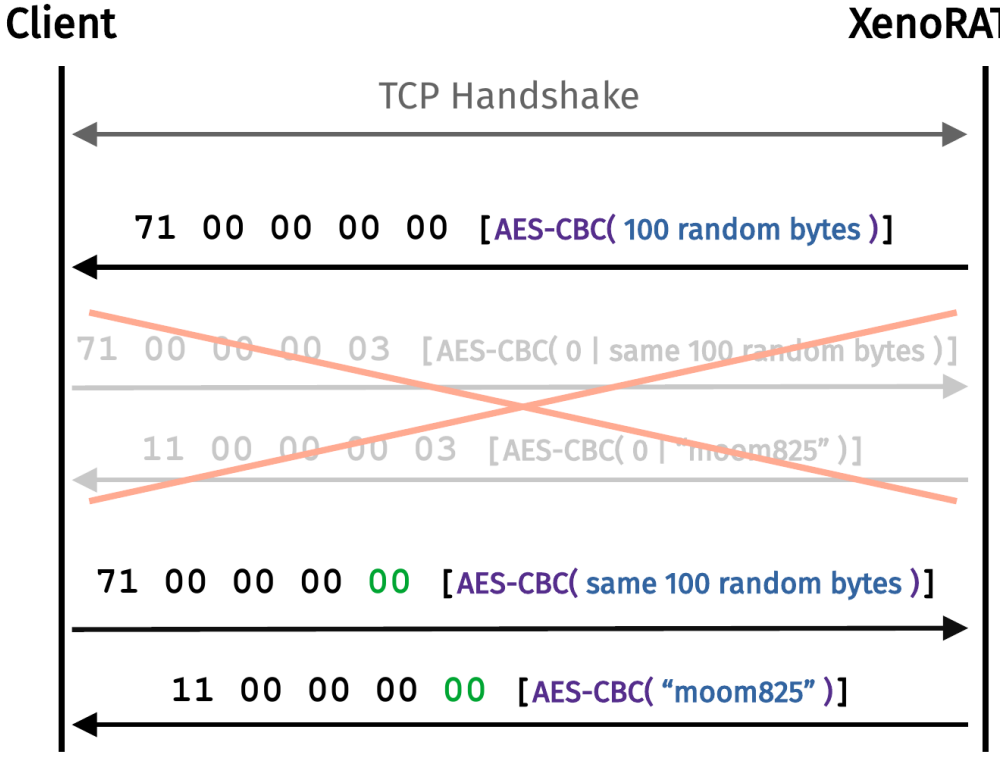
The Low-Hanging Fruit

In order to exploit this handshake procedure to detect XenoRAT C2 servers, first of all, the characteristic byte patterns of the initial server message can be used as an indicator. This is, `71 00 00 00 00 [0x70 bytes of encrypted data]` (due to the mentioned fact that compression doesn’t occur here). This would translate to a Censys query of `services.banner_hex:"7100000000*"`. And looking at the results of that query, we already find numerous XenoRAT C2 servers, that Censys doesn’t label as such. However, this query is not very specific, false positives are possible.

Going one step further

Technically, having received the initial server packet, a scanner could not proceed with the handshake, because, since the AES key is not known, it wouldn't be able to construct a valid packet of the second packet format (with the additional single zero in the plaintext), which a real implant would answer with. And, if the client's response is not valid, the server closes the connection.

However, weirdly and luckily for us, due to an unused code path inside the XenoRAT C2 server, it also possible to answer with the respective 100 random bytes using the first packet format. A real implant wouldn't do this, since it exclusively uses the second packet format but it works nevertheless. Because the same AES key is used for both communication directions, a client actually doesn't need to know the AES key for sending a valid response, that contains the 100 random bytes. **Hence, any client can respond validly by just sending the received initial server packet back as is**. After the C2 server has received this packet, which constitutes a valid response, it sends the further packet containing the encrypted string "moom825". And, as the fifth byte of the client packet is not 0x3, `doProtocolUpgrade` is not changed to true and the C2 server sticks with the first packet format.



Technique to identify XenoRAT C2 server

By that, we didn't only achieve that the XenoRAT C2 server sends a second characteristic packet to the client, giving us a considerably more accurate indicator. With that second server packet, we can additionally determine, whether the default password "1234" ist used, since the plaintext ("moom825") is known. When this default password is used, the respective packet bytes are `11 00 00 00 00 dc db 8d 8b 56 4b f3 37 ae 1a e8 c3 b7 2e 8c 8c` and one is able to further impersonate a real implant, since the AES key is then known and the client thus can construct valid messages.

With the following code snippet, we can realize the described detection method:

python

```
 import socket
import binascii
import sys
import re

regex = re.compile(rb"\x71\x00\x00\x00\x00[\x00-\xff]{112}", re.DOTALL)

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((sys.argv[1], int(sys.argv[2])))
server_packet_1 = sock.recv(1280)
print("initial packet received from server:", binascii.hexlify(server_packet_1))
if re.search(regex, server_packet_1):
         print("packet matches initial xeno rat server message!")
print("\nsending the same packet back...")
sock.send(server_packet_1)
server_packet_2_hex = binascii.hexlify(sock.recv(1280))
print("response:", server_packet_2_hex)
if (server_packet_2_hex == b"1100000000dcdb8d8b564bf337ae1ae8c3b72e8c8c"):
         print("xeno rat server uses the default password '1234'!")
         print("=> the AES key is 03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4")
         print("and the IV is 00000000000000000000000000000000")
```

Example

Querying Censys with `services.banner_hex:"7100000000*"`, I got an interesting match for the domain `daddeln[.]eu` at port 4444, which is the default port for XenoRAT. Probing the server with the code snippet from above yields the following output:

```
 python3 xenorat_connector.py daddeln.eu 4444
initial packet received from server: b'71000000007589af3a2b3415e2b73f1d88443ba63f7fdb521e99662dc83f22c445f1a5aa0
packet matches initial xeno rat server message!

sending the same packet back...
response: b'1100000000dcdb8d8b564bf337ae1ae8c3b72e8c8c'
xeno rat server uses the default password '1234'!
=> the AES key is 03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4
and the IV is 00000000000000000000000000000000
```

In this example, the RAT operator didn’t even bother to change the default password. Hence, we would be able to impersonate an implant.

Source: https://axmahr.github.io/posts/xenorat-detection/