NanoCore RAT Hunting Guide

By John F

Published: 2022-09-14 · Archived: 2026-04-06 00:44:12 UTC

Analysis and tools for hunting NanoCore command-and-control

9 min read

Aug 30, 2022

NanoCore is a prevalent RAT (Remote Access Trojan) which is used by threat actors to spy on victims and provide remote access to target computers. CISA identified NanoCore as a top malware strain of 2021 due to its prevalence among cyber criminals and potential to cause damage. I have researched NanoCore's command-and-control framework and I wrote this blog post as a guide for hunting-down and blocking NanoCore.

Table of Contents

Press enter or click to view image in full size

Background and Functionality

In 2013, Taylor Huddleston (arrested in 2017) wrote NanoCore using the .NET framework. Despite its age, the RAT remains popular among cyber-criminals and malware-as-a-service markets because of its ease of use for non-technical threat actors and various plugin features. NanoCore is widely used today with ANY.RUN Trends documenting it as the 8th most-common malware strain in July 2022.

As a RAT, NanoCore is well-suited for providing initial access, stealing information, and spying on victims. Historically, NanoCore's remote access and spyware capabilities have been used to attack businesses, stalk victims, and conduct espionage for nation-state groups[1]. There are multiple NanoCore plugins which can be purchased (or stolen/cracked) from cyber-crime forums to add new features and capabilities — though NanoCore's base payload is already capable of:

- accessing files
- executing programs
- stealing saved passwords
- logging keystrokes
- and surveilling webcams

NanoCore payloads are primarily controlled from a GUI installed on the threat actor's computer. The GUI functions as both a payload builder and command-and-control panel. Using the GUI to create and operate NanoCore payloads requires limited technical knowledge as there is no shell involved and the interface provides guiding documentation. Once built and deployed, a threat actor can control single agents or an entire collection of bots from the panel.

Press enter or click to view image in full size

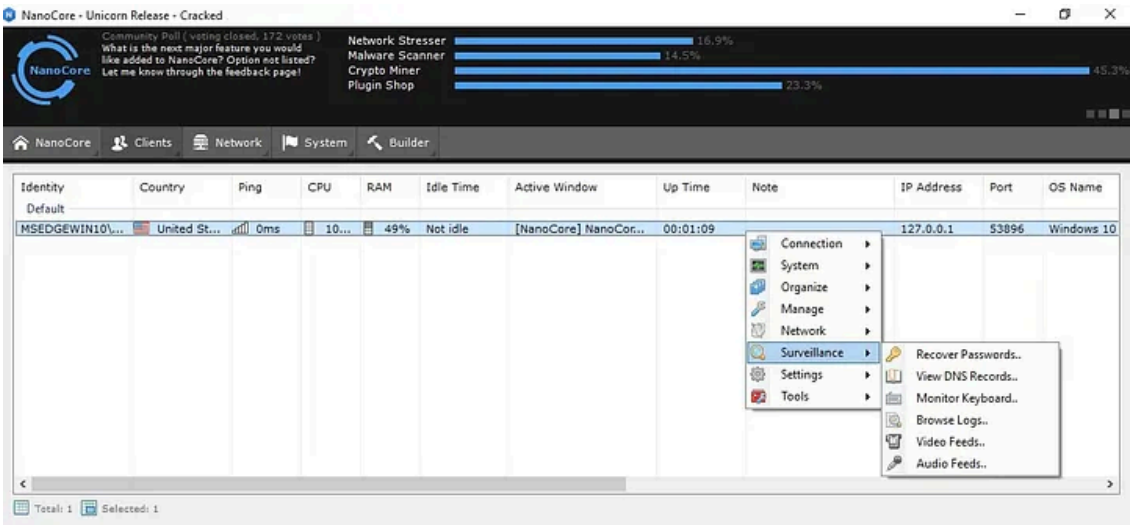

Figure 1: Experiments with NanoCore platform included builder configurations and traffic analysis.

I downloaded cracked NanoCore platforms and used them to conduct experiments inside virtual machines. It's disturbing to see the "community" which forms around malware-as-a-service capabilities…

Command-and-Control Analysis

At start-up, a NanoCore payload will query the domain name of its assigned C2 (Command-and-Control) server and then attempt to use the answered IP address for all of its communications[2]. NanoCore payloads are configured with 8.8.8.8 as their primary DNS server and 8.8.4.4 as a backup. Some payloads are also configured with primary and secondary C2 domains — though in practice, many threat actors just list the same C2 twice.

A NanoCore payload will then send an 'introduction' message to its C2 server with basic context about the payload configuration and the infected target host. Following an acceptance message from its C2 server, NanoCore will then begin sending a 'heartbeat message' to the C2 server in order to let the server know that the payload is still active and to request new tasks. NanoCore will encode these heartbeat messages to the C2 server as *0x00000600*. Experiments show that NanoCore will send these heartbeat messages almost immediately after processing the C2's response — leading to a stream of communication (unlike other RATs/agents which may only communicate once every several minutes). The generic heartbeat messages are interspersed with commands and requests from the C2 server itself. Most often, these messages are automated requests for status updates though they will also include commands initiated by the threat actor.

Press enter or click to view image in full size

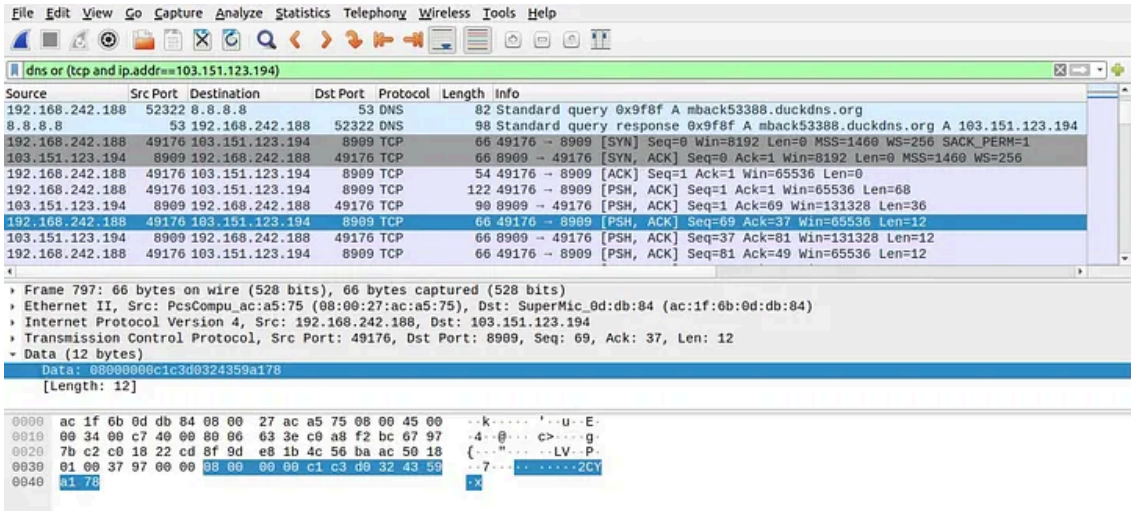

Figure 2: PCAP of server DNS request, C2 introduction message, and encrypted heartbeat

The content of the messages sent from NanoCore to its C2 server will be encrypted using a custom protocol over TCP. NanoCore uses DES to encrypt the contents of its messages (*curiously, the DES key and IV have the same value*). The length of the message ciphertext is then appended to the front of the TCP data payload and sent to the C2 server. I wrote a CyberChef recipe to help explain NanoCore's custom protocol and decrypt its messages.

Press enter or click to view image in full size

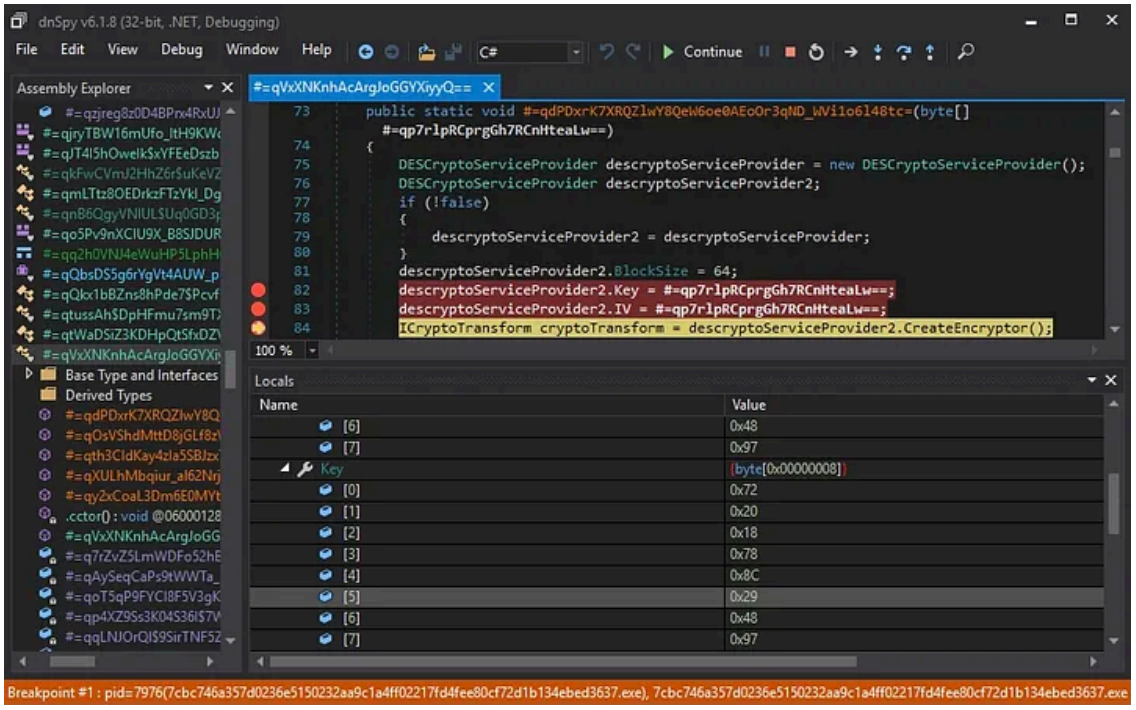

Figure 3: Debugging of NanoCore's method for message encryption — including IV/Key values

For more analysis of NanoCore's actual code, check-out the awesome reverse engineering conducted by

and .

Network Defenses

Signature-Based Detection

NanoCore's encryption of its C2 communications makes PCAP analysis and signature mapping more difficult. The custom protocol which NanoCore implements, however, inadvertently creates patterns that can be leveraged to develop network signatures[3]. There are a few DES Keys/IVs that have been hard-coded into NanoCore strains and are commonly used in different payloads — even those appearing to belong to unique threat actors. Accurate network signature rules for tools such as Snort can be written to monitor for NanoCore's heartbeat message, as encrypted by these common keys.

```
# NanoCore 'heartbeat' signature (0x00000600) encrypted by common key
alert tcp any any -> any any (msg:"NanoCore 'heartbeat' signature";flags:PA;dsize:12;content:"|08 00
```

Several less-popular NanoCore strains have begun changing their DES Keys/IVs — making the encryption of the heartbeat message difficult to predict. The command-and-control messages from these rarer strains can still be detected by taking-advantage of NanoCore's custom protocol. In the protocol, the first 4 bytes of a TCP payload are used to encode the length of a message and the possible lengths are limited due to the protocol's reliance on DES. NanoCore's custom C2 messaging protocol can still be detected regardless of Key/IV by searching the first 4 bytes of TCP data and matching the value to the message length.

```
# NanoCore command-and-control custom message protocol
alert tcp any any -> any any (msg:"NanoCore DES length 08";flags:PA;dsize:12;content:"|08 00 00 00|"
alert tcp any any -> any any (msg:"NanoCore DES length 40";flags:PA;dsize:68;content:"|40 00 00 00|"
```

By implementing these network signature rules into a monitoring tool such as Snort or Suricata, you will be alerted to an infected host sending NanoCore's 'introduction' to the C2 server, the encrypted 'heartbeat', and all subsequent communications which use its custom message protocol.

Get John F's stories in your inbox

The rules have been released publicly in my GitHub repository NanoCore-Hunting. Additionally, I strongly encourage you to monitor for ***incoming*** *C2 traffic as well as outgoing. I have worked with cloud services, small businesses, and others in the past to help them take-down malicious C2 servers that they were unknowingly hosting. It is better to handle hosting abuse before your infrastructure is used to harm others.*

Infrastructure Blocking

Traditional network blocking solutions can of course be used to deny access to NanoCore C2 servers — though these methods require constant updating. With a DNS sinkhole deployed[4], DNS requests from NanoCore can be answered with an incorrect IP address to redirect NanoCore from its real C2 server — though this redirection is only possible if the DNS sinkhole is configured with the payload's domain name (for more information, see Palo Alto's explanation of DNS sinkholes). Simple firewalls can also be deployed to block[5] connection attempts to known NanoCore IP addresses but again, this blocking solution only works if the C2 server's address is known and added to the blocklist.

The following lists are based on the hundreds of recently-active NanoCore C2s I verified while researching the RAT. The lists likely do not include all active NanoCore C2s but they should block a decent amount of connections.

IP-list.txt domain-list.txt Indicators-of-Compromise.csv

Those of you familiar with my SarlackLab project know that I track malicious infrastructure — not to look for specific IOCs but to gather intelligence and get a better picture of the Internet threat landscape.

Meta-Analysis of IOCs

I set my malware analysis lab to collect and analyze thousands of NanoCore samples throughout the Spring and Summer of 2022. By analyzing network traffic generated during sample detonation and cross-referencing apparent C2 traffic with malware configuration extractors, I confirmed the existence of hundreds of NanoCore C2 servers.

Domain Patterns

I analyzed trends among the NanoCore C2 servers I had identified and uncovered patterns for server domains as well as suspicious hosting providers. Almost all of the domain names I found are legible and do not appear to be high-entropy or randomized. Many of the Fully Qualified Domain Names (FQDNs) have the same second-level domains (see Figure 4) — of which most appear to be tied to dynamic record and redirection services.

Press enter or click to view image in full size

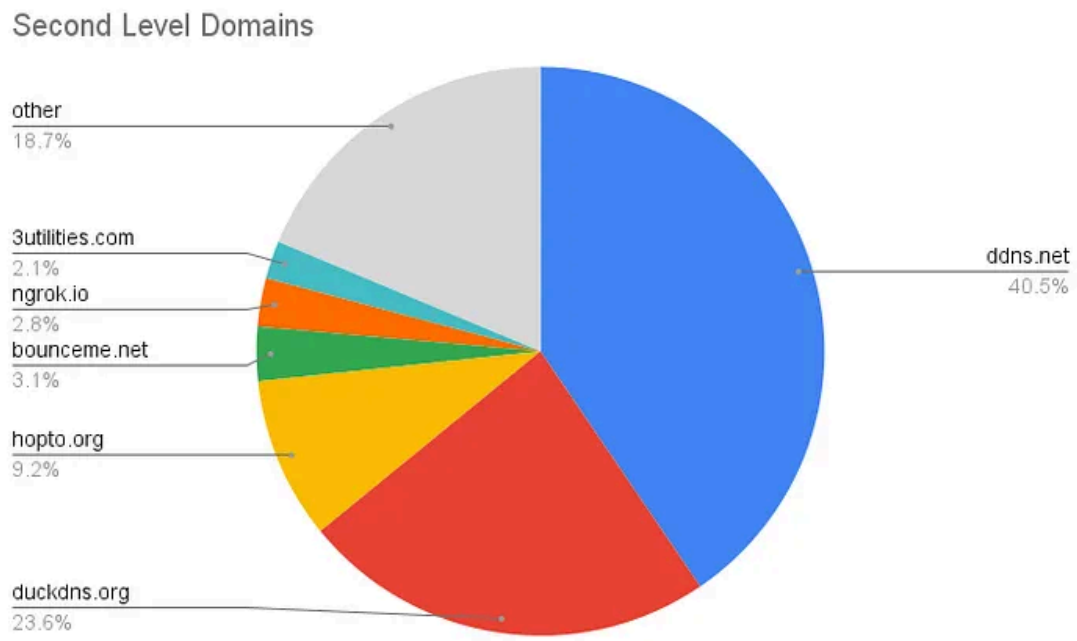

Figure 4: Second-Level Domains of NanoCore C2 Domain Names

The second-level domains ddns[.]net, duckdns[.]org, hopto[.]org, bounceme[.]net, and 3utilities[.]com appeared in almost 80% of all NanoCore FQDNs. The listed second-level domains are tied to Dynamic DNS services which their FQDNs appear to be utilizing. The service ngrok[.]io, however, is a distributed reverse proxy that is commonly used as a network tunnel for hosting gaming servers or file shares — though it also has a history of abuse.

```
Dynamic DNS (abbreviated as DDNS) is built upon common Domain Name System (DNS) technology but uses
```

I have used DNS sinkholes to great effect in denying access from hosts to various C2 servers — regardless of how threat actor modify their DNS or DDNS records. Personally, I recommend sinkholing[4] DNS requests to all subdomains of the identified infrastructure and authoritative DDNS servers, then only allowing specific FQDNs as-needed[6]. By configuring such a sinkhole, many NanoCore payloads will be unable to resolve the IP address of the C2 server — even if the payload is attempting to connect to a novel subdomain, not yet identified as an IOC.

Part 2 — TLP:GREEN

There is a second part to this command-and-control meta-analysis which is labeled TLP:GREEN. If you are interested, contact me via Twitter and I will send you a copy of the research.

Hunting Summary

NanoCore is a powerful remote access trojan that remains popular among cybercriminals today. To hunt-down NanoCore command-and-control communications in your network, look for the signature of its custom messaging protocol (see published rules). You can of course attempt blocking individual IPs and sinkholing specific domains — but I recommend considering the overall trends of malicious traffic on the Internet and denying access to commonly malicious infrastructure. Please contact me on Twitter, if you have any feedback or wish to contribute.

See Also

Other work on NanoCore analysis (technical deep-dives). Config extractor. Other places to watch.

- Full break-down of IOCs available at https://github.com/Abjuri5t/Hunting-NanoCore/
- CyberChef recipe https://gchq.github.io/CyberChef/#recipe=…

Footnotes

[1] According to research compiled by MITRE ATT&CK, groups back by nation-states have been attributed to the use of NanoCore.

[2] In my research, I found a couple of NanoCore payloads which used hard-coded IP addresses. The overwhelming trend among threat actors, however, appears to be relying on domain names because the hosting IP addresses may change.

[3] I work for Vectra AI as a Network MDR Analyst. I honestly believe that network metadata and AI models (ya, I said it) are the best indicators of malicious activity in a network. That being said, signatures and IOCs are useful as part of a complete balanced ~~breakfast~~ defense. I wrote a Vectra custom model based on these IDS signatures which Recall customers can install.

[4] When configuring your DNS sinkhole, remember to re-direct all DNS requests to your local server. Otherwise, NanoCore's DNS requests will just go to 8.8.8.8 and the sinkhole will fail.

[5] Make sure you log these connection attempts as well and consider setting an alert for them. Just because you have blocked a C2 server does not mean that you have completely managed the threat.

[6] Check your SIEM and/or firewall logs before applying these restrictions. Depending on how locked-down your network is, you may accidentally block an important service that happens to be using ngrok or similar.
