How DCRat (AKA Dark Crystal) Works

CYBERSECURITY / 05.09.22 / The BlackBerry Research and Intelligence Team

Update 05.27.22*: An unknown APT group is targeting Russian government entities with at least four separate spear-phishing campaigns since the beginning of the Ukraine conflict. Source: Security Affairs.*

In the murky underworld of Russian crimeware, DCRat seems to be a bit of a dark horse. Unlike the well-funded, massive Russian threat groups crafting custom malware to attack universities, hospitals, small businesses and more, this remote access Trojan (RAT) appears to be the work of a lone actor, offering a surprisingly effective homemade tool for opening backdoors on a budget. In fact, this threat actor's commercial RAT sells at a fraction of the standard price such tools command on Russian underground forums.

DCRat (also known as DarkCrystal RAT) is a commercial Russian backdoor that was first released in 2018, before being redesigned and relaunched a year later. Notably, this threat appears to have been developed and maintained by a single

across. The price for this backdoor starts at 500 RUB (less than 5 GBP/US$6) for a two-month subscription, and occasionally dips even lower during special promotions. No wonder it's so popular with professional threat actors as well as script kiddies.

This price range is a curious feature, as it makes it seem like the author is not particularly profit-driven. It could be that they're simply casting a wide net, trying to get a little money from a lot of maliciously minded people. It could also be that they have an alternative source of funding, or perhaps this is a passion project rather than their main source of income.

Peering Deeper into the Dark Crystal

DCRat's modular architecture and bespoke plugin framework make it a very flexible option, helpful for a range of nefarious uses. This includes surveillance, reconnaissance, information theft, DDoS attacks, as well as dynamic code execution in a variety of different languages.

The DCRat product itself consists of three components:

- A stealer/client executable
- A single PHP page, serving as the command-and-control (C2) endpoint/interface
- An administrator tool

The administrator tool is a standalone executable written in the JPHP programming language, an obscure implementation of PHP that runs on a Java virtual machine. As with the examples discussed in our previous whitepaper discussing exotic programming languages used by malware writers, JPHP offers some potential benefits for making mischief.

As a programming language, JPHP's target audience is primarily entry-level developers who make cross-platform desktop games. The ease of use, as well as the portability of its code, suits this purpose well. The malware author may have chosen this format because it's not particularly well-known, or they might have lacked programming skills in other, more mainstream languages.

According to the JPHP documentation, this implementation "compiles PHP sources to Java Virtual Machine (JVM) bytecode, which can then execute on the JVM." The JPHP project also provides a dedicated, Russian-language integrated development environment (IDE) called DevelNext. This IDE was used to develop the DCRat administrator tool, as well as some of the early versions of the DCRat client.

Location data available in public GitHub profiles indicates the core contribution team behind JPHP are overwhelmingly based in the Commonwealth of Independent States (CIS), an intergovernmental organization made up of twelve post-Soviet countries. The DCRat author's decision to use JPHP may have stemmed from either an assumed level of trustworthiness, or simply from a belief that obtaining support for issues or enhancements related to the JPHP framework would have been easier to establish due to their shared familiarity with the Russian language.

Examining the DCRat Build

The DCRat client binary – meant for delivering to victim's machines – is written in .NET. Earlier versions were written in JPHP, like the administrator tool. This was likely done to streamline and optimize the client component. JPHP is rather slow, as it runs on the JVM. And the distributed malware is much smaller, since it doesn't have to include all the JPHP libraries.

DCRat is built around a modular architecture that incorporates a plugin framework. Affiliates can generate their own client plugins, which can be downloaded and used by subscribers. (We've included a list of the current plugins in the "Plugins" section, later on in this blog.)

The RAT currently seems to be under active development. The administrator tool and the backdoor/client are regularly updated with bug fixes and new features; the same applies to officially released plugins.

many high-profile attacks, including campaigns against U.S. government institutions in 2021.

A detailed analysis of the DCRat client was published by Mandiant in May 2020. Just days after this report was released, the malware author shifted distribution of the RAT to a new domain. It's clear that cybercriminals are becoming more aware of publicity from media and the security community, and they're getting used to making swift changes in response to this unwanted exposure.

It's worth noting that there is a second open-source RAT that also goes by the name DcRAT, which can be found in GitHub repository of user "qwqdanchun." This is most likely a completely unrelated project. While it doesn't bear many code similarities to DCRat, it may have been an inspiration for – or inspired by – the threat.

DCRat Offering

The DCRat bundle, its plugins, plugin development framework, and additional tools are currently hosted on crystalfiles[.]ru. These components have been moved there from their previous location at dcrat[.]ru. The crystalfiles website features a simple interface, as seen in Figure 1 below, and it serves only as the download point for the RAT. It has no additional information or resources for potential or existing clients.

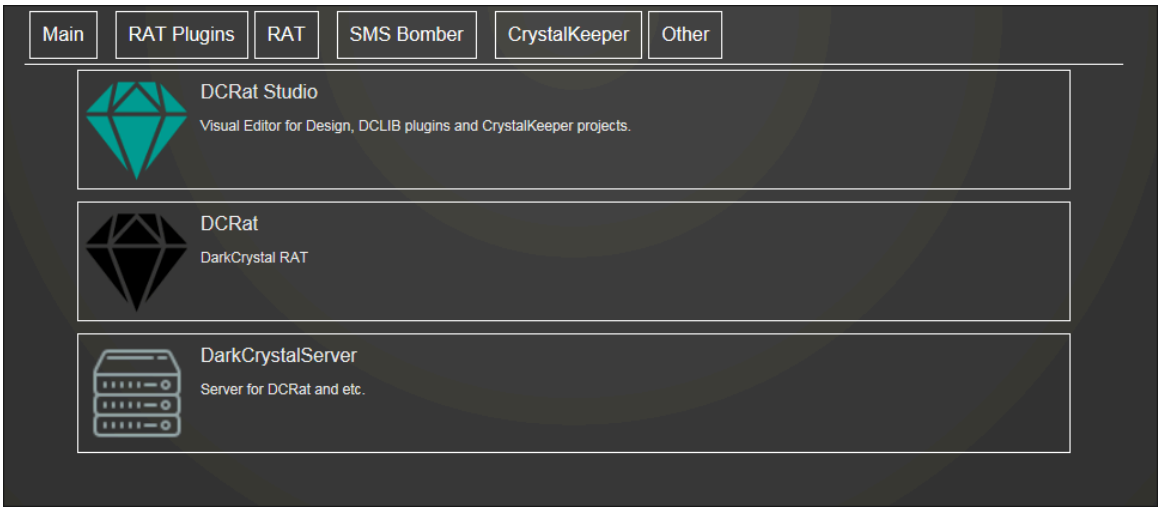


Figure 1 – Download links for DCRat components at crystalfiles[.]ru

All DCRat marketing and sales operations are done through the popular Russian hacking forum lolz[.]guru, shown in Figure 2, which also handles some of the DCRat pre-sales queries. DCRat support topics are made available here to the wider public, while the main DCRat offering thread is restricted to registered users only.

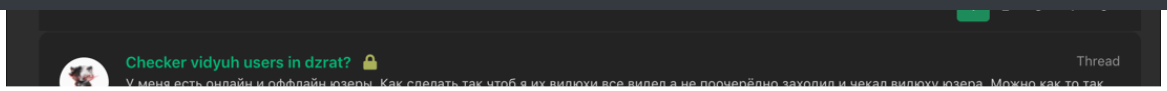


Figure 2 – lolz[.]guru forum – discussions about DCRat

It's possible that the RAT is also sold on other restricted-access forums or on the dark web. The DCRat archives have been spotted on other URLs, and they've been shared through Discord instant messaging. The most common file name for distribution, across different versions of the RAT, seems to be "1ac770ea1c2b508fb3f74de6e65bc9c4.zip."

All news and updates for DCRat are announced through a dedicated Telegram channel, as seen in Figures 3 and 4 below. At the time of writing, the channel had almost 3k subscribers.

Figure 3 – DCRat Telegram page providing news and updates

Besides the DarkCrystalRAT Telegram account, there are also two Telegram bots: one for processing sales requests ("DCRatSeller_bot"), and one for technical support ("CrystalSupport_bot").

The latest prices for DCRat licenses (excluding any temporary discounts) are:

- 500 RUB / US$5 for two-month license
- 2200 RUB / US$21 for a year
- 4200 RUB / US$40 for a lifetime license

Figure 4 – DCRat Telegram announcing discounts and price specials

The Author

While the DCRat developer posts as Кодер ("Coder") on the lolz[.]guru forum (as shown in Figure 5), their Telegram handle is "@boldenis" and their GitHub username is "boldenis44" (based on a resource link buried in the DCRat source code shown in Figure 6). They must have used the latter name on lolz[.]guru at some point, as some users still refer to them as such. They list their email address as crystalcoder[at]exploit[.]im. The date of birth and address listed on their profile shown in Figure 5 below are most likely fake.

The lolz[.]guru forum profile indicates the developer is Russian and works alone.

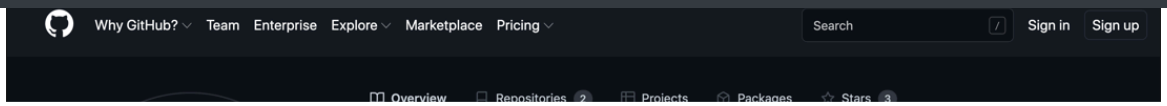
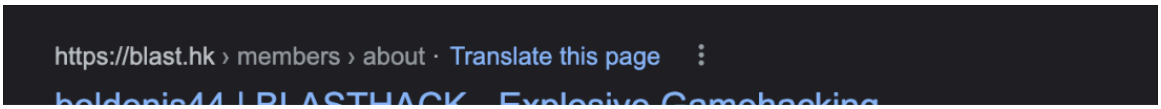


Figure 6 – GitHub page under the same account name as DCRat author

"Boldenis44" also has accounts on game-hacking forum blast[.]hk, the Russian Minecraft server gamai[.]ru, as well as on the Russian dark marketplace DarkNet[.]ug, shown in Figure 7.



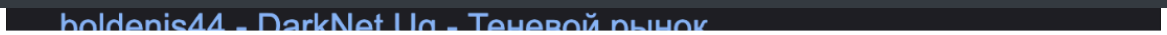


Figure 7 – Search results for "boldenis44," author of DCRat

There is also a "Darkcrystal Rat" profile on VKontakte, a Russian social network at vk[.]com (dcrat_1994), but it's unclear if it belongs to the same person as boldenis44 / Coder. This profile page is shown below in Figure 8.

Figure 8 – VKontakte profile page for Darkcrystal Rat

The description in Russian translates roughly to "I steal data, I work on ru, uk and what?" It's not entirely clear what this means, though it's likely they're bragging about stealing data from Russia, the UK, and possibly other countries.

The photo in this profile comes from a 2014 German hacker movie called "Who Am I: No System is Safe." This photo has recently been changed – the cached version of this website shows an image (see Figure 9) that is a relatively popular depiction of a hacker, and the Russian sentence that somewhat cryptically translates to: "I drive SS into Dark."

Figure 9 – Google Cache view of an earlier version of the Darkcrystal Rat profile

Another malware writer, claiming to be the author of a notorious RAT called njRAT, recently changed their profile photo to the same frame from "Who Am I," as shown in Figure 10.

Figure 10 – Facebook page of njRAT author, featuring the same avatar as Darkcrystal Rat profile

This is most likely a coincidence, as the njRAT profile is written by someone who speaks Arabic, not Russian.

There was another profile on the VKontakte site that has been spotted mentioning the crystalfiles[.]ru URL, as shown in Figure 11, which was for Rodion Balkanov (Родион Балканов): https[:]//vk[.]com/bagyuvix. However, this account has since been removed and is no longer available.

Figure 11 – Google view of VKontakte page mentioning DCRat distribution URL

The Timeline

Although the DCRat project appears to have started several months in advance, a larger scale marketing campaign took place in September 2019, when the Telegram channel was created and the dcrat[.]ru domain registered. Shortly after this, the RAT got significantly redesigned to support plugins in a bespoke format.

The next major release came in May 2020 (version 3.0), followed by version 4.0 in March 2021. In between major releases, the RAT got smaller updates and bug fixes on a very regular basis, hinting that the author was highly engaged with his creation during this timeframe, as shown below.

- July 31, 2018 – Кодер ("Coder") profile created on lolz[.]guru forum
- Sept. 1, 2019 – Telegram channel called DarkCrystalRat created
- Sept. 2, 2019 – dcrat[.]ru registered (see Figure 12)

Figure 12 – dcrat[.]ru domain whois information

- Sept. 4, 2019 – Introduction of a bespoke plugin format: DCLIB
- Nov. 19, 2019 – Redesign of the administrator tool
- May 12, 2020 – Mandiant publishes analysis of DCRat client
- May 27, 2020 – crystalfiles[.]ru registered; distribution shifts to the new domain

Figure 13 – crystalfiles domain whois information

- May 30, 2020 – Version 3.0 released
- Oct. 2020 – Release of DCRat Studio, a bespoke platform that allows third-party developers to design plugins
- March 18, 2021 – Version 4.0 released
- Dec. 31, 2021 – Limited-time 50% discount on all types of licenses, as a New Year's Eve deal
- March 6, 2022 – Due to devaluation of ruble, pricing changed from rubles to dollars at an exchange rate of US$1 = 100 RUB
- March 28, 2022 – Limited-time price discount for two-month, one-year, and lifetime licenses to $5, $19, and $39, respectively.

New plugins and minor updates are announced almost every day.

DevelNext compiles the PHP program into a Java bytecode, which can then be executed on the JVM.

According to its GitHub page, the IDE is still in the beta stage, and it's only available in the Russian language at this point. In the past, we've seen very few malware samples written in JPHP, because the executables it produces are both exceptionally large and slow to run.

One example of malware using this IDE is a rudimentary backdoor called IceRAT, discovered in early 2020. This malware targeted Russian-speaking victims by installing crypto-mining software on their endpoints. An older example is one that was written for OSX as part of a campaign targeting Jaxx cryptocurrency wallets, which was discovered in 2018.

Contents of the Archive

The administrator tool comes as a ZIP archive with the following structure:

File name	Description
DCRat.exe	Admin launcher (created using Launch4j wrapper)
dcrat_updservice.exe	Admin updater tool
updatelauncher.bat	Script that executes dcrat_updservice.exe
Notify.wav	Audio file with notification sound (2.5 sec)
data/	Location of helper utilities
design/	Contains DeleteAll_legacy.json file
lib/	Location of all the Java modules of the builder
plugins/	Used to store downloaded plugins; by default, contains only a test plugin
profiles/	Empty directory used to store user's saved profiles

The **lib** directory is home to the main builder module, together with several legit JPHP modules that the builder depends on.

SHA256 hash	Description
9967ea3c3d1aee8db5a723f714fba38d2fc26d8553435ab0e1d4e123cd211830	JSON module
6014d44d8f7da00f03db051b3dcea9a03ec3837977118c69a4512ef558a6df2a	Main builder module
cf4068ebb5ecd47adec92afba943aea4eb2fee40871330d064b69770cccb9e23	GUI module
5b37e8ff2850a4cbb02f9f02391e9f07285b4e0667f7e4b2d4515b78e699735a	JPHP core module
4aef566bbf3f0b56769a0c45275ebbf7894e9ddb54430c9db2874124b7cea288	zend module
d637e3326f87a173abd5f51ac98906a3237b9e511d07d31d6aafcf43f33dac17	jfoenix module
c25d7a7b8f0715729bccb817e345f0fdd668dd4799c8dab1a4db3d6a37e7e3e4	javafx module
2d43eb5ea9e133d2ee2405cc14f5ee08951b8361302fdd93494a3a997b508d32	Google gson module
15f36830124fc7389e312cf228b952024a8ce8601bf5c4df806bc395d47db669	PHP module

434e57fffc7df0b725c1d95cabafdcdb83858ccb3e5e728a74d3cf33a0ca9c79	XML module
0f26584763ef1c5ec07d1f310f0b6504bc17732f04e37f4eb101338803be0dc4	JPHP SDK module
4bec0794a0d69debe2f955bf495ea7c0858ad84cb0d2d549cacb82e70c060cba	javafx module
03ead999502aefbf1380bd2e9c4a407acb7a92a7b2fe61f6995aba3fca85efd4	objectweb asm module

Builder's entry point is specified in <**main_module**>.jar/.system/application.conf and points to dct/forms/MainForm.phb.

```
# MAIN CONFIGURATION
app.name = DCRat2.0
app.uuid = fabb4b64-bb3a-4418-a495-a0e669188d81
app.version = 1

# APP
app.namespace = dct
app.mainForm = MainForm
app.showMainForm = 1
app.fx.splash.autoHide = 0
```

The **data** directory contains a bespoke compiler for producing the client executable, a bespoke EXE obfuscation tool, a commercial .NET protection tool called .NET Reactor, and compression utilities WinRAR and UPX.

SHA256	File name	Description
d0680ac62e94f953df031533acd0acb718ad8494f938d84198c655507709e5df	7zxa.dll	Legit 7zip DLL
914cca033fc8ca52830a21b5dca55263cee1e74ab5571702906ee9c25aedafd7	DCRAC.exe	DCRat EXE obfuscator
812cd4b5e80bc4e83a2e01a6f3fb24346ecf57dcaf8ff6fc3e55a2a6b953da23	DCRCC.exe, DarkCrystalRATCSharpCompiler.exe	DCRat compiler
b11ad1adfa96eacf5f18cf87785884947a6d35a1baebf4f20f16402b04d5109f	Default.SFX	Part of WinRAR
a0b6bb521e52a99abf5ac1017302da014d37296619078d42d9edf5d86d137f63	NCC2.dll	Part of .NET Reactor
38274608d5a4b53ec22f8099f798ba46ce0ed41db65a33dfb3853f0dbf849f6f	NCC3.dll	Part of .NET Reactor
c41cd461470ff3c936e225cea37e5190cb06e3cd70a3d76ca8e5d3aceead5493	NCCheck.dll	Part of .NET Reactor
770d7b5e40ed9b0aff5d0e3fc2ccf9ba10d4925d3441f38b71a35bd26e6e8d98	Rar.exe	Part of WinRAR, signed

		signed
db28575f61b1adc88a28ae51ce3b00226e4974ca60894896e414ea408c6ff9fe	RarExt64.dll	Part of WinRAR, signed
ca08ed8423afda4b41757a1f3adf4f855732dc0628fe2ea5d8a96b13f56b9f84	WinCon.SFX	Part of WinRAR, signed
2293fe261d5c6f5f2a33004b11f068037677b7aa5a6f792031e31555f31f0d69	Zip.SFX	Part of WinRAR
83445595d38a8e33513b33dfc201983af4746e5327c9bed470a6282d91d539b6	dnlib.dll	DNLib - .NET assembly reader/writer library
e817802f166662a7df0b144571354d74b10e34d120f91ae9d84ca3ba925241c6	dotNET_Reactor.Console.exe	Part of .NET Reactor
78684aea83b1a5c402a87ba0ce2e7ad5b0338462cc804e97369203ce53d29834	dotNET_Reactor.exe	Part of .NET Reactor
5981e508e89c65c445fca892e91b8ec39b1d8563804d0999d963d640aa592444	enc.vbe	Script used to encode VBS scripts
d634cde09d1aa1320a1d4c589d35d306f8350129faf225b2bca394128c2c4442	upx.exe	UPX 3.96 Windows 32-bit
1317d70682bd11e5d320af850d6ecbb5a70c200d626ec7bf69c47566894db515	wRar.exe	Part of WinRAR, signed

PHB file format

Instead of Java class files, the JPHP JAR archives are composed mainly of PHB files.

PHB is a custom file format used exclusively by JPHP. PHB files are simply archives that contain uncompressed, unencrypted Java class files and a PHB header. Each Java class file is preceded by a class file header, containing information such as module name, method names, PHP file path, and the class file length.

Class files can be extracted with the following Python script, then decompiled using tools such as JAD or jd-gui.

```
import os
import sys
import struct

in_file = sys.argv[1]
out_dir = os.path.splitext(in_file)[0] + "_extracted"
```

```
with open(in_file, 'rb') as f:
 buf = f.read()
 magic = b'\xCA\xFE\xBA\xBE'
 offsets = [i for i in range(len(buf)) if buf.startswith(magic, i)]
 count = 0
 for of in offsets:
  file_name = os.path.splitext(in_file)[0] + "_" + str(count) + ".class"
  f.seek(of - 4)
  class_len = struct.unpack('>i', f.read(4))[0]
  file_data = f.read(class_len)
  with open(os.path.join(out_dir, file_name), "wb") as f2:
   f2.write(file_data)
  count += 1
```

PHB file structure (example):

```
1C 9A 4A 92       PHB signature
01 33 53 D3
00 00 00 00
00 00 00 33
00 33             len of the following string
44 3A 5C 49       string "D:\IdeaProjects\DCRat2.0\src\dct\forms\MainForm.php"
[...]
00 00 00 2D
00 2D             len
24 70 68 70       "$php_module_mba8a6a7b4b0144048b64e6456cd9fb81"
[...]
00 01
FF FF FF FF
FF FF FF FF
00 07             len
55 6E 6B 6E       "Unknown"
[...]
00 00 00 00
00 00 03 4F       number of class files
00 00 00 36       start of class file header #1
00 36             len
24 70 68 70       "$php_module_mba8a6a7b4b0144048b64e6456cd9fb81_closure0"
[...]
00 00 00 00
00 00 00 00
FF FF FF FF
00 00
FF FF FF FF
FF FF FF FF
00 01
00 00 00 00
00 00 00 08
00 08             len
```

```
00 00 00 08
00 08             len
5F 5F 69 6E       "__invoke"
76 6F 6B 65
01 00 00 00
39 00 00 00
18
00 33             len
44 3A 5C 49       "D:\IdeaProjects\DCRat2.0\src\dct\forms\MainForm.php"
[...]
00 00 0A 98       len of class file #1
CA FE BA BE       start of class file #1
[...]
00 00 00 36       start of class file header #2
00 36             len
24 70 68 70       "$php_module_mba8a6a7b4b0144048b64e6456cd9fb81_closure1"
[...]
00 00 00 00
00 00 00 00
FF FF FF FF
00 00 FF FF
FF FF FF FF
FF FF 00 01
00 00 00 00
00 00 00 08
00 08             len
5F 5F 69 6E       "__invoke"
76 6F 6B 65
00 00 00 08
00 08             len
5F 5F 69 6E       "__invoke"
76 6F 6B 65
01 00 00 00
40 00 00 00
18
00 33             len
44 3A 5C 49       "D:\IdeaProjects\DCRat2.0\src\dct\forms\MainForm.php"
[...]
00 00 0A 9B       len of class file #2
CA FE BA BE       start of class file #2
[...]
```

Licensing

The DCRat administrator tool, shown below in Figure 14, prevents unauthorized use through a series of online license checks. Once these checks succeed, the administrator interface becomes available.

Figure 14 - Administrator tool license checks preventing unauthorized use

The checks consist of HTTPS queries to the hardcoded domain dcrat[.]ru.

Peer Validation

The first validation check transmits a random 64-character value, hashed and Base64-encoded prior to transmission. The response from the C2 server must contain the same value, and it must be similarly hashed and encoded to be considered valid. This exchange provides rudimentary peer validation, ensuring the administrator tool is communicating with a genuine DCRat license server.

Subscriber Validation

A second HTTPS request authenticates the computer on which the administrator tool is running, as shown in Figure 15. A handful of host properties are collected to generate a unique fingerprint. This is transmitted to dcrat[.]ru and will (presumably) match against a valid subscriber entry.

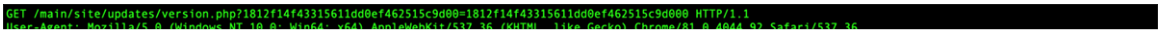


Figure 15 – License validation HTTPS queries to dcrat[.]ru domain

Kill Switch

The administrator tool also performs an unusual final HTTPS check to a public resource hosted on GitHub, under the personal space of "boldenis44." The query and response functions have a global "kill switch," as shown in Figure 16. At the DCRat author's discretion, flipping this switch would render all instances of the DCRat administrator tool unusable, irrespective of subscriber license validity (so much for that "lifetime license"!).

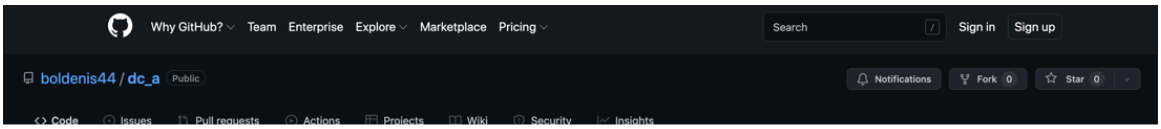


Figure 16 – GitHub-hosted master kill switch; still active

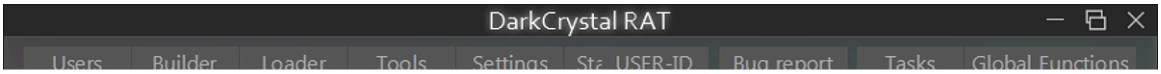
This kill switch feature was found in separate administrator tool builds dated mid-2021 and early 2022.

Administrator Functions

The administrator tool allows a subscriber to take the following actions:

- Login to an active C2 server
- Issue Tasks to registered client installations
- Generate builds of the Loader and/or Client
- View and query installation statistics
- Submit bug reports to the DCRat author

Login needs to be performed to an active C2 server hosting the backend PHP, as shown in Figure 17.



Login parameters follow an obscure syntax:

- http://<server>/@<reversed_base64_PHP_pagename_minus_php_suffix>
- password

Fake News?

For reasons that are not entirely clear, the DCRat author implemented a function that displays a randomly generated number of "Servers working" and "Users online" that are meant to appear as statistics in the background of the administrator tool. It could be that they are trying to make their tool appear more popular, or that they just didn't know how to implement an accurate counter and have employed a pseudo-counter in the meantime as a placeholder.

Admin Functions

Following authentication, the administrator tool begins polling the C2 for details of connected and infected hosts.

Functions are grouped using tabs, as shown in Figure 18:

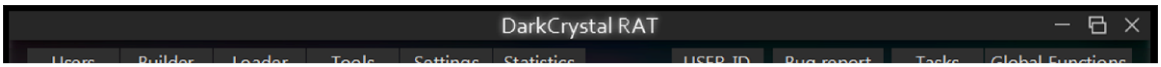


Figure 18 – Administrator tool major functions tab

Users

This tab lists the active/registered installations of DCRat client running on infected hosts. The list is updated using a periodic poll to the C2.

Builder

This tab is where the threat actor can configure (and generate) builds of the DCRat client executable. In the analyzed version of the administrator tool, the "core" of the client is downloaded from the dcrat[.]ru domain as a Base64 string, becoming input for "DCRCC.exe."

Figure 19 – Administrator tool configuration page for client runtime settings

These are the available parameters for configuration:

- Network:
  - Specifies a list of primary and secondary C2 hosts (transport protocols are limited to HTTP/S).
- Protect: (shown in Figure 19)
  - Optional obfuscation of generated client binaries using .NET Reactor
  - Mutex name to use during execution – by default it's a random 20-character alphanumeric string preceded by DCR_MUTEX prefix
  - DCR_MUTEX-<20_ALPHANUM_RAND>
  - Disable Windows Task Manager via Registry entry (see IoC)
  - Specify (spoof) PE creation time stamp
  - Launch delay in seconds
- Plugins:
  - Configure and enable DCRat plugins
- Installation:
  - Path for unpacking modules when DCRat client runs
  - Persistence mechanism to use
  - First start command script to use
  - Tag value to appear on hosts running DCRat build (i.e., campaign ID)
  - Auto functions – functions to start automatically after launch:
    - Stealer
    - Keylogger
    - Uninstall (auto-delete)
  - Force Admin – try to force admin rights on launch
  - Build Cache Storage
- Build:

- PE file icon

Loader

Configure and build a DCRat loader binary. Support is provided for a range of stackable "Actions" combining to determine runtime behavior:

- Download file
- Execute file
- HTTP request
- CMD Script
- Wait
- Message Box

Tools

Provides file upload and Netscape to JSON cookie converter.

Settings

Configure Builder settings:

- Change GUI background image
- Automatically poll C2 for connected (infected) hosts/installs
- Show notifications

Statistics

Canned reports to query DCRat client installations (country, Windows version, etc.)

USER-ID

We cannot confirm at present what this function is for. It's possible that this is a direct remote control/terminal client to an infected host.

Bug Report

Submit bug report to DCRat maintainer(s).

Tasks

Configure Tasks to be executed on one or more DCRat clients. Tasks can be Saved (exported) or Loaded (imported) from text file. Tasks are stored as a reversed Base64 string.

Global Functions

Configure Tasks to be performed on all registered DCRat clients.

DCRat Client

In this section we review the features of the DCRat client (stealer) and the DCRat Loader. Runtime behavior for both is configured using the DCRat administrator tool.

Client Loader

The administrator tool provides a function to generate a DCRat "Loader" executable. In the version we analyzed, generation of a loader in DLL format was not supported. It's conceivable the author could add this support in newer builds.

The behavior of the Loader when executed is configured via one or more canned "Actions," as shown in Figure 20. A typical build might be a combination of "Download File," "Wait" and "Execute File," which would silently pull down a file and then

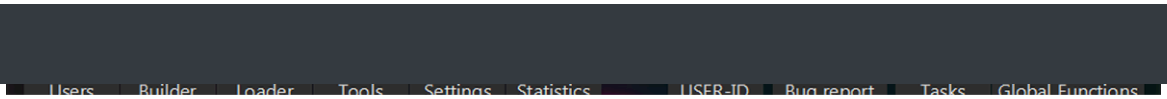


Figure 20 – Runtime tasks for DCRat loader, configured using the Aadministrator tool

The source code for the Loader is embedded within the administrator tool as a series of Base64 strings that decode to reveal C# source code. Code for the executable is selected based on the Actions chosen by the user. The bundled "DCRCC.exe" generates the executable.

If selected, the generated executable will be protected using DotNET Reactor:

> "-control_flow_obfuscation 1 -flow_level 9 -resourceencryption 1 -stringencryption 1 -suppressildasm 0 -all_params 1 -obfuscate_public_types 1 -exception_handling 0"

Persistence

Persistence for DCRat is limited to common Windows "autorun" locations:

1. Registry, using HKLM\Software\Microsoft\Windows\CurrentVersion\Run
2. Registry, using HKCU\Software\Microsoft\Windows\CurrentVersion\Run
3. Registry, using HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon (REG_SZ: "Shell")
4. Scheduled Task /ONLOGON
5. Scheduled Task /sc minute RandomMinMax(5,15)

The client executable copies itself to the System drive root (e.g., C:\) using the name of a randomly chosen running process, excluding "svchost.exe."

Config

DCRat's config is embedded in the client binary as a Base64-encoded string resource. It has a JSON format and contains C2 URLs, a tag, a mutex name and a few execution options, as well as plugin-specific configuration options for included plugins.

Name	Type	Description
H1	string	Primary C2 URL
H2	string	Secondary C2 URL
TAG	string	A tag specified at build time (e.g., victim ID, campaign ID, etc.)

		string
DBG	bool	Debugging on/off
BCS	int	Build cache storage size
AUR	int	*Exact use unknown; controls file rename/persistence behaviour*
AS	bool	Auto-stealer on/off
AK	bool	Auto-keylogger on/off
AD	bool	Auto-uninstall on/off
PLUGINCONFIGS	object	Plugin-specific configuration options

Below is an example config found in a sample distributed through the Prometheus TDS:


```
{
  "H1": "http[:]//co44089.tmweb[.]ru
/9rsk8lug9peq4f23cjhyo3fz2q7j81vhnvil6c6tjdc7adzbia1ki04d9p65b5wfe4ronb0rtm/4vsyc5bajheyp1gt5i63igklh15828uwuwsek0x0p9frsqy1l2boc3l936aratwc7jddw2djz
  "H2": "http[:]//co44089.tmweb[.]ru
/9rsk8lug9peq4f23cjhyo3fz2q7j81vhnvil6c6tjdc7adzbia1ki04d9p65b5wfe4ronb0rtm/4vsyc5bajheyp1gt5i63igklh15828uwuwsek0x0p9frsqy1l2boc3l936aratwc7jddw2djz
  "TAG": "GFN",
  "MUTEX": "DCR_MUTEX-bQ2or3bMKAwvUmZaLKHY",
  "DBG": false,
  "BCS": 0,
  "AUR": 1,
  "AS": true,
  "AK": true,
  "AD": false,
  "PLUGINCONFIGS": {
    "MessageOnStartConfig": {
      "caption": "GFN hacker",
      "text": "Wait 10 minutes",
      "icon": "Information",
      "buttons": "OK",
      "uniq": "chpf05oqbupjilp1ccxqb65xf"
    },
    "XMRigMinerCFG": {
      "SavePuth": "C:/WindowsDefender/RunShell.exe",
      "Gate": "xmr.pool.minergate.com:45700",
      "UserName": "Fuzzii2739@gmail.com",
      "Password": "x",
      "DopArguments": "--donate-level=1 --pause-on-battery",
      "CPUPriority": "0",
      "cpumaxthreadshintn": "25",
      "mode": "light"
    }
  }
}
```

As part of initial registration, the DCRat client reports a range of host attributes to its C2. This information is determined using a combination of WMI, .NET-provided instrumentation classes, and Windows registry queries:

- Host computer name
- Host username
- Windows product/version
- Tag value (embedded; campaign id)
- Is Administrator
- Video card name(s)
- CPU Product/Vendor
- Local, network, removable drive labels
- Has microphone
- Installed webcam(s)
- Active Window text
- Country, City, Lat/Long (geoip)
- Antivirus product(s) installed
- Firewall product(s) installed
- BIOS manufacturer
- Motherboard manufacturer
- CPU Vendor
- Physical memory
- Network interfaces (IP, WiFi/Ethernet)
- .Net version installed

```
using (ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher("SELECT * FROM Win32_PnPEntity WHERE (PNPClass = 'Image' OR PNPClass =
  'Camera')"))
{
```

Figure 21 – DCRat stealer WMI query to identify webcam devices as part of host fingerprinting

All HTTPS transactions use a random User Agent, picked from an embedded array of 12:

```
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36",

"Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:94.0) Gecko/20100101 Firefox/94.0",

"Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0) Gecko/20100101 Firefox/95.0",

"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Safari/537.36",

"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Safari/537.36",

"Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0",

"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.55 Safari/537.36 Edg/96.0.1054.34",

"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Safari/537.36 Edg/95.0.1020.53",

"Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36",
```

"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36 Edg/96.0.1054.29",

"Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:91.0) Gecko/20100101 Firefox/91.0"

Stealer Functionality

The Stealer functions of DCRat are pre-configured using the administrator tool "Builder." Stealer "Tasks" define the sequence of operations carried out during theft of stored information:

DCRat can steal from the following sources (including those pictured in Figure 22):

- Browser cookies
- Browser stored passwords
- Browser stored form content
- Browser history
- Stored credit cards (via Windows DPAPI & Chrome SQLite Database)
- Telegram
- Steam account
- Discord tokens
- FileZilla credentials
- Screenshots
- Keylogger
- Clipboard contents
- Sysinfo

```
new Struct11("GPUName", Class76.Bbl_
new Struct11("CPUName", Class76.cTy_
new Struct11("Webcams", Class44.smet
new Struct11("Microphones", Class44.
new Struct11("BIOS", Class76.P26_Get
new Struct11("LANIP", Class76.smetho
new Struct11("Antivirus", Class76.sm
new Struct11("Firewall", Class76.ia2
new Struct11("Motherboard", Class76.
new Struct11("RAM", Class76.XSa_GetP
new Struct11("Screens", Class44.ubR(
new Struct11("SteamPath", Class44.R5
```

The Stealer component is also capable of running bespoke plugins, making it extensible to accommodate information malware authors find on specific targets.

Denial of Service

The DCRat Stealer contains primitive, multi-threaded code to perform different forms of DOS attacks – including HTTP(S) POST, UDP and TCP – to a specific host and endpoint combination.

Delay Tactics

Common to many malware families, DCRat employs the use of Windows command line tools to perform execution delays. Associated with the execution of DCRat client are invocations of the Windows command line tool for time service configuration, w32tm. When configured with suitable command line arguments, as shown in Figure 23, it can act as a delay mechanism. In the case of DCRat, arguments are passed that act as 10 second delays. Coincident instances of w32tm in endpoint XDR could be a possible, albeit somewhat weak, signal of DCRat client execution:

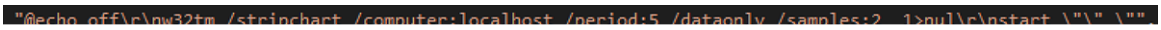


Figure 23 – Delay commands used when self-terminating

Plugins

Plugins can be designed by third-party developers with the use of a dedicated IDE called DCRat Studio. Official plugins are available to download from crystalfiles[.]ru (as shown in Figure 24) and their functionality includes data exfiltration/credential stealing, system manipulation, and cryptocurrency mining.

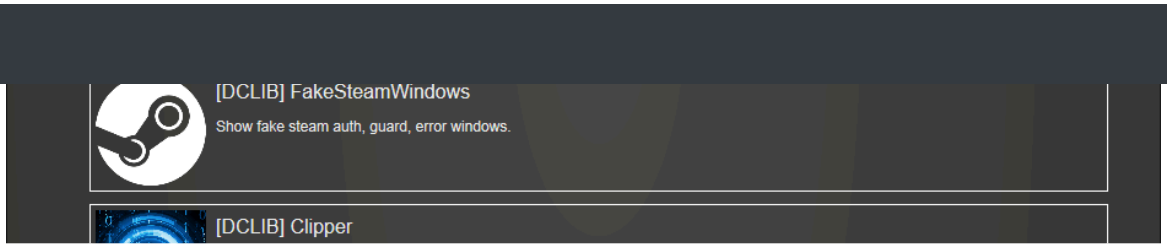


Figure 24 – Plugins available to subscribers for download

To harness the power of crowd-sourced development and to encourage an ecosystem of plugins that target different information stores, DCRat subscribers have access to a list of supported third-party plugins. The precise inner workings of each plugin are unknown, but the name of each does provide an indicator of function:

- AutoKeylogger (deprecated)
- AntiVM (merged with AntiAnalysis)

- RunOnce
- DesktopGrabber
- StartupPlus
- AntiKiller
- AntiSNG
- BlockInput
- MessageOnStart
- ClipboardLogger
- RegEditor
- FileSearcher
- FileGrabber
- TitleKiller (deprecated)
- ProcessKiller
- CryptoStealer
- TelegramNotifier
- AntiAnalysis
- Clipper
- CountryBlackList
- VPNGrabber
- ForceAdmin
- SystemRestorePointsCleaner
- UserPingCounter
- ActiveWindowNotifier
- FakeSteamWindows
- Discord notifications (third party)
- IgnorTags (third party)
- Kryptex Miner (third party)
- XMRig (third party)

Conclusions

The biggest, flashiest threat groups might get their name in lights, but they aren't necessarily the cybercriminals that keep security practitioners up at night. The scary, cutting-edge threats that come out of those advanced and well-funded threat groups do occasionally cause headaches for those of us who aren't guarding state secrets or ridiculous amounts of money. But miscreants with too much time on their hands can often cause just as much hassle.

Generally speaking, you get what you pay for, even in malware. If you pay a pittance for something, you would be wise to expect it to be less functional or poorly supported. But DCRat seems to break that rule in a way that's deeply perplexing.

This RAT's code is being improved and maintained daily. If the threat is being developed and sustained by just one person, it appears that it's a project they are working on full-time.

There are certainly programming choices in this threat that point to this being a novice malware author who hasn't yet figured out an appropriate pricing structure. Choosing to program the threat in JPHP and adding a bizarrely non-functional infection counter certainly point in this direction. It could be that this threat is from an author trying to gain notoriety, doing the best with the knowledge they have to make something popular as quickly as possible.

While the author's apparent inexperience might make this malicious tool seem less appealing, some could view it as an opportunity. More experienced threat actors might see this inexperience as a selling point, as the author seems to be putting in a lot of time and effort to please their customers.

Indicators of Compromise (IOCs)

DCRat Stealer; Self Preservation; Windows Registry changes:
HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System
REG_DWORD: "DisableTaskMgr":1

DCRat Stealer; Persistence; Windows Registry:
HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon
REG_SZ: "Shell": "explorer.exe, %STEALER_EXE_PATH%"
HKCU|HKLM\Software\Microsoft\Windows\CurrentVersion\Run: <STEALER_EXE_PATH>

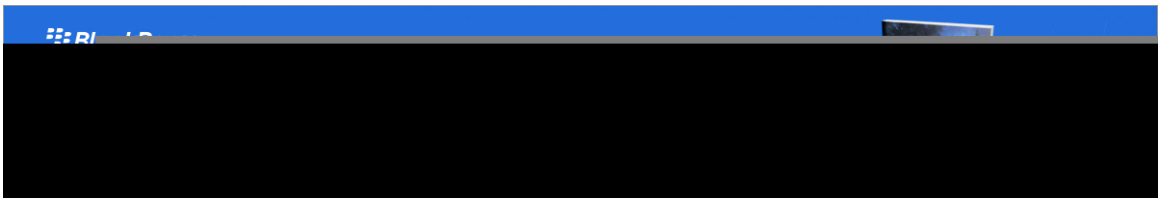
DCRat Stealer; Persistence; Windows Scheduled Tasks:
schtasks.exe /create /tn <STEALER_EXE_NO_EXTENSION> /sc ONLOGON /tr <STEALER_EXE_PATH> /rl HIGHEST /f
schtasks.exe /create /tn <STEALER_EXE_NO_EXTENSION> /sc minute /mo <RND_MIN5_MAX15> /tr <STEALER_EXE_PATH> /f

DCRat Stealer; Host Fingerprint; WMI Queries:
SELECT * FROM AntivirusProduct: displayName
SELECT * FROM FirewallProduct: displayName
SELECT * FROM Win32_BIOS: Manufacturer
SELECT * FROM Win32_BaseBoard: Manufacturer, SerialNumber
SELECT * FROM Win32_Processor: Name
SELECT * FROM Win32_ComputerSystem: TotalPhysicalMemory
SELECT * FROM Win32_VideoController: Name, AdapterRAM
SELECT * FROM Win32_PnPEntity WHERE (PNPClass = 'Image' OR PNPClass = 'Camera')

DCRat Stealer; Host Fingerprint; Windows Registry:
READ: HLKM\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full\Release
READ: HKLM\SYSTEM\ControlSet001\Control\Class\ {4d36e968-e325-11ce-bfc1-08002be10318}\ <SUBKEY_1..SUBKEY_N>\{AdapterString,DriverDesc,qwMemorySize}

DCRat Stealer; Runtime; Mutex (Default format, if not overridden):
DCR_MUTEX-<20_ALPHANUM_ULCASE_RAND>

DCRat Builder/Admin Tool; C2 Network Traffic:
DNS + HTTPS: dcrat[.]ru, crystalfiles[.]ru



About The BlackBerry Research and Intelligence Team

The BlackBerry Research and Intelligence team is a highly experienced threat research group specializing in a wide range of cybersecurity disciplines, conducting continuous threat hunting to provide comprehensive insights into emerging threats. We analyze and address various attack vectors, leveraging our deep expertise in the cyberthreat landscape to develop proactive strategies that safeguard against adversaries.