2024 DUBLIN

2 - 4 October, 2024 / Dublin, Ireland

A WILD RAT APPEARS: REVERSING DINODASRAT ON LINUX

Anderson Leite & Fabio Marenghi

Kaspersky, Brazil

anderson.leite@kaspersky.com
fabio.marenghi@kaspersky.com

ABSTRACT

DinodasRAT is a previously publicly undocumented multi-platform remote access trojan created in C++ and offering a range of capabilities that enables the malicious actor to surveil and harvest sensitive data from a target's computer. In October 2023, the *Windows* version was discovered by *ESET* attacking governmental entities in Guyana.

We then started examining telemetry data and were able to identify the *Linux* version of the same RAT about a month later. Some samples indicate that this RAT was previously observed in a joint research effort by *Kaspersky* and *TeamT5* around 2021, linking it to the LuoYu APT group and also known as XDealer.

In this paper we'll discuss the analysis of the *Linux* implant used by attackers, how it achieves persistence in *Linux* systems, and the C2 protocol used. The main highlights of this *Linux* threat are:

- DinodasRAT is a remote access trojan infecting systems worldwide, with both *Windows* and *Linux* versions. The *Linux* versions were primarily designed to target *Red Hat*-based distributions and *Ubuntu*.
- The implants come with the capability of utilizing proxies to communicate with the C2 server. For encryption of communication between the implant and the C2, DinodasRAT uses `libqq qq_crypt` library functions that use the Extended Tiny Encryption Algorithm (XTEA) in CBC mode within a custom network protocol.
- The trojan hides access to the file by modifying its timestamp, creates profiles in the /etc/ directory, and is able to fully control *Linux*-based machines.

INTRODUCTION

DinodasRAT is a potent remote access trojan (RAT) affecting systems worldwide, with versions targeting both *Windows* and *Linux*. The *Linux* variants are particularly designed to infiltrate *Red Hat*-based distributions and *Ubuntu*. As a post-exploitation tool, the initial access vector of DinodasRAT remains unknown. However, attackers must have some level of access to the server, typically achieved through methods such as exploiting vulnerabilities or stealing SSH credentials.

```
if ( argc != 3 )
{
  daemon(0, 0);                    // Persitence with SystemD or SystemV
  InstallPersistence();
  pid = getpid();
  get_executable_path(&executable_path);
  sprintf_0(&command, "%s d %u", executable_path, pid);
  new_cmd = command;
  while ( 1 )
  {
    system(new_cmd);               // Reexecute as a child process
    sleep(5u);
  }
}
root = IsRoot();
argv_ref = *argv;
RunningAsRoot = root;
v7 = strlen(argv_ref);
std::string::assign(&arguments, argv_ref, v7);
sscanf(argv[2], "%u", &parent_pid);
mal_main();
```

Figure 1: DinodasRAT main.

The backdoor operates through several distinct phases. Initially, it executes directly without any arguments. It then transitions to running as a daemon, allowing it to operate quietly in the background.

To ensure persistence on the infected system, it utilizes SystemV or SystemD startup scripts. The backdoor subsequently reruns with arguments related to the current (parent) process ID. Finally, it waits for the child process to execute.

VICTIM REGISTRATION DATA

Before establishing contact with the C2 server, the backdoor gathers information about the infected machine. Notably, the attackers do not appear to be interested in collecting user-specific data, indicating that this backdoor is primarily designed for infecting *Linux*-based servers. The gathered information typically includes:

- Date of the infection
- MD5 hash of the 'dmidecode' command output
- A random number as an ID.

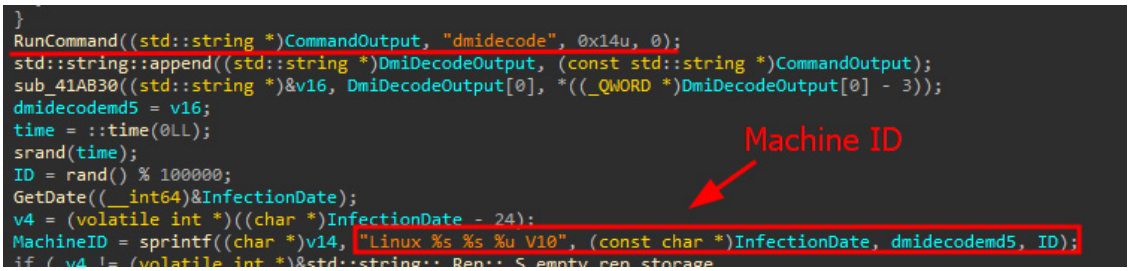

Figure 2: Machine unique identifier generation.

This set of information is enough to create a unique identifier for the server. As 'dmidecode' returns hardware-specific information about a machine, taking an MD5 hash of this information provides an effective way to establish a unique identifier.

PERSISTENCE

DinodasRAT in its *Linux* version takes advantage of the two versions of service management to establish persistence on an affected system: SystemD and SystemV.

When the malware starts, a function is called that will get the type of distribution that the victim is running. Currently, there are two flavours of distros that the implant targets based on the readings of '/proc/version' – *Red Hat* and *Ubuntu 16/18* – although, technically, it could infect any distro that supports one of two versions of init scripts.

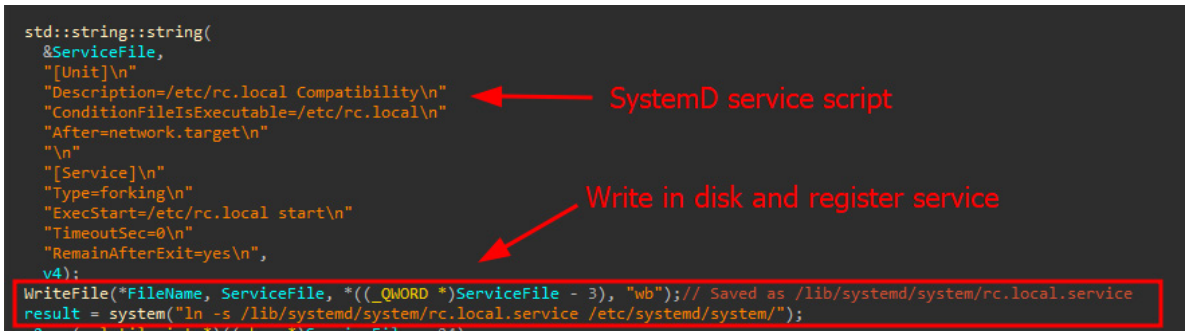

Figure 3: SystemD service script.

Once the system is recognized, it will install a suitable init script that provides persistence for the RAT. In the case of *Ubuntu*, it will create a SystemD script located at '/lib/systemd/system/rc.local.service' (with the description '/etc/rc.local Compatibility'), which executes a script located at '/etc/rc.local'. This script will be executed once the network setup is done, and it launches the backdoor.

SERVER PROFILE

To maintain a record of all victim information, DinodasRAT stores it in a hidden file located at '/etc/.netc.conf'. This file will contain a configuration file structure. The basic file layout only includes the 'imei' field, used as an identifier by the operators:

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

Figure 4: Infected server profile.

The following entries are possible for the profile:

Section entry	Key	Description
para	imei	ID of the infected machine
para	va	New C2 IP address
para	checkroot	'1' if the current privilege level is root, otherwise '0'
para	ptype	Proxy address
para	mode	Defines how the malware should behave in determinate routines, it can also hold other values such as '64' or '32' to indicate the bits of the current OS.

This information is loaded using a modified version of the header-only library named `inifile-cpp`. The original header code still contains assertions that suggest the actual name of the DinodasRAT project, `ntfsys_v3`.

```
__assert_fail(
  "section != __null && strlen(section)",
  "/home/soft/ntfsys_v3/src/inifile.cpp",
  0xCDu,
  "static int IniFile::read_profile_string(const char*, const char*, char*, int, const char*, const char*)");
```

Figure 5: Assertion that leaks the project name and path.

Not only does DinodasRAT store this information in a hidden file, but whenever access for read/write operations is required, it will subsequently alter the stat structure to reflect an older date, which is static and hard coded into the code. This tactic is employed to evade detection by forensic and integrity monitoring tools. However, knowing the altered, static timestamp might serve as an indicator of compromise in incident response.

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

New access date

Figure 6: Access time modification of any file read/write.

In addition to replacing the access time of this file upon execution, DinodasRAT also modifies the access time on the executable *itself* after its execution.

C2 COMMUNICATION & ENCRYPTION

After a victim profile is created, DinodasRAT initiates several threads responsible for various tasks, including monitoring the connection state, retrieving the list of currently logged-in users, and sending this information to the C2 server.

One of the created threads continuously checks if there are any users connected to the infected machine. DinodasRAT executes the `who` command, which retrieves the list of users logged in at a given moment. If there are active connections besides the malicious operator, it will block any attempts to establish connections with the command-and-control (C2) server, closes all sockets and wait for the user to log out. This pattern is repeated for each connection attempt.

```
if ( (unsigned __int8)IsTCP((char *)&socket_type + 64, 1)
  && !(unsigned __int8)CheckUserLogin((__int64)&g_checkuserlogin) )
{
  return 0;
}
else
{
  C2Connection = 0;
  GetC2Address((__int64)&Dinodas, &arg->sa_family10, &arg->dword18);
  thread_lock(&arg->ppthread_mutex0);
  ResolveC2IP(
```

User is logged in, abort the connection

No one are logged, continue the connection

Figure 7: DinodasRAT user-logged-in check.

Communication in DinodasRAT can occur via either TCP or UDP protocols. This configuration can be adjusted by the operator. By default, DinodasRAT will search for this entry in the profile configuration, on the key 'mode' of the entry 'param'. If it is not '1' or is not available, UDP will be used as the default protocol. As this field also holds the current machine bits (32 or 64), it will cause most of the connections to fall into UDP.

The network packet consists of a header and an optional payload field. The header includes the checksum (CRC-32) of the entire packet. Among the various fields in the network packet, the vital ones are: TimeStamp, Packet type, and Encrypted Payload Size. A simplified version of the network packet is depicted in Figure 8.

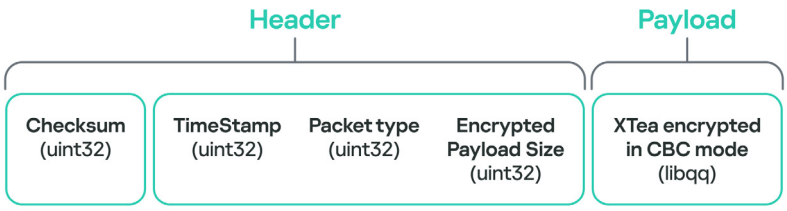

Figure 8: Simplified network packet structure.

The payload section stays empty in certain packet types, like heartbeat packets, which simply inform the C2 that an infected machine is online. When the C2 responds to this packet, the malware will send the generated *Linux* UID as an identifier of the machine in an encrypted format together with the privilege level of the current malware execution.

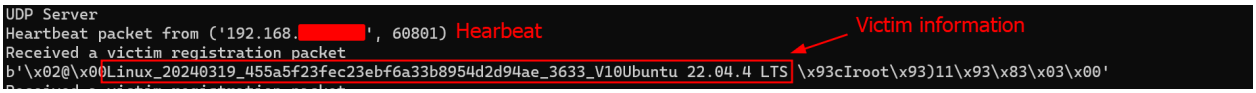

Figure 9: Communication example in a controlled environment.

For encryption of communication between the implant and the C2, along with encryption of data, it uses Pidgin's `libqq` `qq_crypt` library functions for encryption and decryption. This library uses the Extended Tiny Encryption Algorithm (XTEA) as a way to cipher and decipher the data, which makes it fairly easy to port between platforms. This is another similarity with the *Windows* RAT. The *Linux* implant also shares two of the keys used in the *Windows* version:

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

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

BACKDOOR COMMANDS

Although the *Linux* version of DinodasRAT shares some commands with its *Windows* sibling, the majority of the commands are new in comparison. Here's a list of the commands seen in this sample that are known, as well as new ones being introduced:

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	DealChgIp	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	This command controls an available service. If 1 is passed, 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 data 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)
0x2E	UninstallMm	Uninstall implant and delete any artefacts on the system

CONCLUSION

In this research, we highlighted different aspects of this RAT. We have seen an overview of how the *Linux* variant of DinodasRAT creates a custom configuration for each infected individual. We have also seen the steps it takes to avoid

being recognized by different file integrity detection tools, and the mechanisms it uses to avoid being seen while active in the infected systems.

These, along with a custom network protocol and use of encryption, makes the *Linux* version of DinodasRAT an efficient implant utilized in APT campaigns.

REFERENCES

[1] Tavella, F. Operation Jacana: Foundling hobbits in Guyana . WeLiveSecurity. 5 Octobter 2023. https://www.welivesecurity.com/en/eset-research/operation-jacana-spying-guyana-entity/.

[2] Google. libqq-pidgin - default. https://code.google.com/archive/p/libqq-pidgin/source/default/source.

[3] Wikipedia. Tiny Encryption Algorithm. https://en.wikipedia.org/wiki/Tiny_Encryption_Algorithm.

[4] Chang, L.; Niwa, Y.; Ishimaru, S. LuoYu: Continuous Espionage Activities Targeting Japan with the new version of WinDealer in 2021. Japan Security Analyst Conference 2022. https://jsac.jpcert.or.jp/archive/2022/pdf/JSAC2022_7_leon-niwa-ishimaru_en.pdf.