Dissecting ValleyRAT: From Loader to RAT Execution in Targeted Campaigns

picussecurity.com/resource/blog/dissecting-valleyrat-from-loader-to-rat-execution-in-targeted-campaigns

Sıla Özeren Hacıoğlu November 5, 2025

CREATED ON November 05, 2025

ValleyRAT is a multi-stage Windows remote-access trojan (RAT) first observed in early 2023 and now linked to focused campaigns targeting Chinese-language users and organizations.

Operators deliver staged components (downloader → loader → injector → RAT) via phishing or trojanized installers, and rely on in-memory decryption and living-off-the-land execution (notably MSBuild.exe) to remain stealthy. The malware includes targeted checks, for example, a registry "kill-switch" for regionally common apps such as WeChat and DingTalk, plus multiple UAC bypasses, token manipulation, and routines that disable or terminate locally prevalent AV/HIPS. That combination of targeted execution logic, aggressive privilege escalation, and AV disruption means responders treat ValleyRAT as a high-confidence indicator of a targeted intrusion rather than commodity malware.

In this blog we will analyze the ValleyRAT loader and the RAT itself, highlighting the features that set ValleyRAT apart from other families.

What Are Analyzed Malware Samples for ValleyRAT

SHA-256: ae857addc8eb51dbfa7d0a76b19dae7a6f275f7bf1042d1c982aca4f80ce635e (downloader)
SHA-256: 3f7819debdca5df5a6cd50147b51bceba12c5e0f8a6961b1612777080496dde1 (loader)
SHA-256: 190d493255c71f3cebb968c197aeef67c62d597b488c4a0b8cd77751e5999b94 (dropper)

What Is the ValleyRAT Loader Malware?

ValleyRAT first surfaced in early 2023 and has primarily targeted Chinese-language users via phishing lures. It’s implemented as a multi-stage, multi-component .NET family:

- the loader is a .NET executable that contains 3DES-encrypted resources (embedded .NET PE files),
- it decrypts and loads components in memory (loader → injector → payload),
- the injector component uses MSBuild.exe as the execution host (process masquerading / LOLBin usage) and performs process injection to run the final RAT payload in memory.

This layered approach improves stealth (encrypted resources on disk, decryption in memory, and leveraging signed system binaries).

What the ValleyRAT loader code shows (high level)

The decompiled snippets include three important artifacts:

- **A Main() style entry** (decompiled, obfuscated) that assembles obfuscated strings using .Replace, Strings.StrReverse, Unicode escape sequences, and unusual concatenation; it sleeps briefly, copies itself into the Startup folder (Appcustom.exe) and calls into an internal class to execute further logic.
- **A loader routine** (waVUUcaYXhu() in our snippet) that instantiates helper objects from an internal class (here referenced as pITLNz), assembles strings at runtime, passes those values into chained methods A, Bout, C, then ultimately calls Dout(...) with a path to MSBuild.exe.
- **A cryptography helper** (TFGFG... / TRforpleasess...) that implements **TripleDES** decryption: it computes MD5(Encoding.BigEndianUnicode(ikey)), sets the result as the 3DES key, uses **ECB** mode, and decrypts an embedded byte buffer.

Below are the essential pieces extracted from the sample (cleaned for readability):

```
// loader calls into decrypted assembly to perform further decryption/execution
public static void waVUUcaYXhu()
{
   pITLNz.SECONDACTCONFIRM secondactconfirm = new pITLNz.SECONDACTCONFIRM();
   pITLNz.TWOFACTCONFIRM twofactconfirm = new pITLNz.TWOFACTCONFIRM();
   string text = "晚上好呀...".Replace("晚上好", string.Empty);
   string text2 = ".".Replace(".", "0").Replace(";", "L") + Strings.StrReverse("صبح بخير".Replace("صبح بخير", "A"));
   object objectValue = RuntimeHelpers.GetObjectValue(twofactconfirm.A(text, ref text, ref text2));
   object objectValue2 =
RuntimeHelpers.GetObjectValue(pITLNz.Bout(RuntimeHelpers.GetObjectValue(objectValue)));
   object objectValue3 = RuntimeHelpers.GetObjectValue(secondactconfirm.C(
      RuntimeHelpers.GetObjectValue(objectValue2),
      Strings.StrReverse("GDEYFHSYWUDUSJDIAUADJIAJKSJA") + text4_modified
   ));
   string text5 = "##...##".Replace("##", "").Replace(Strings.StrReverse("..."), null);
   pITLNz.Dout(6, RuntimeHelpers.GetObjectValue(objectValue3),
            "None", ref text5,
            Path.Combine(RuntimeEnvironment.GetRuntimeDirectory(), "MSBuild.exe"));
}
```

And the symmetric decryption helper:

```
public static byte[] TFGFGF...(byte[] B, string ikey)
{
   TripleDESCryptoServiceProvider triple = new TripleDESCryptoServiceProvider();
   MD5CryptoServiceProvider md5 = new MD5CryptoServiceProvider();
   byte[] array = md5.ComputeHash(Encoding.BigEndianUnicode.GetBytes(ikey));
   triple.Key = array;
   triple.Mode = CipherMode.ECB;
   return triple.CreateDecryptor().TransformFinalBlock(B, 0, B.Length);
}
```

What Tactics, Techniques, and Procedures (TTPs) Does the ValleyRAT Malware Use?

ValleyRAT is not your typical Remote Access Trojan. Analysis of its latest variants reveals a sophisticated malware characterized by highly targeted execution logic, multi-faceted evasion techniques, and an aggressive focus on Privilege Escalation (PE) and Anti-Analysis capabilities.

This deep dive examines the code-level mechanisms that ValleyRAT uses to establish persistence, bypass system defenses, and prepare for its primary Command and Control (C2) operations.

Kill Switch and Targeted Execution Logic

ValleyRAT begins with an unusual environmental check, serving as a geographical or kill switch. It specifically queries the Windows Registry for the presence of two popular Chinese communication applications: WeChat and DingTalk.

If both registry entries (HKCU\Software\DingTalk and HKCU\Software\Tencent\WeChat) are *not* found, the malware assumes it's running outside its intended target environment, displays a misleading "Error" message box, and terminates.

Additionally, the malware employs an anti-duplicate-instance check by attempting to create a named mutex, L"TEST".

```
// Code snippet showing the initial registry check and termination logic
if ( RegOpenKeyExW(HKEY_CURRENT_USER, SubKey, 0, 0x20019u, &hKey)
&& RegOpenKeyExW(HKEY_CURRENT_USER, v1, 0, 0x20019u, &hKey) )
{
  OutputDebugStringA("RC| BB");
  return MessageBoxA(0i64, "Error", "Error", 0);
}
if ( !IsUserAnAdmin() && CreateMutexW(0i64, 0, L"TEST") && GetLastError() == 183 )
  ExitProcess(0xFFFFFFFF);
```

The Aggressive Pursuit of Privilege Escalation

If the initial checks pass, ValleyRAT immediately focuses on gaining administrative access, employing multiple techniques to bypass Windows **User Account Control (UAC)**.

1. Multi-Vector UAC Bypass (T1548.002)

The malware employs a combination of file and registry manipulation targeting known Windows executables:

- CompMgmtLauncher.exe/Event Viewer: ValleyRAT drops shortcut (.lnk) files, CompMgmtLauncher.lnk and eventvwr.lnk, in a user-writeable location (e.g., Startup folder). It then manipulates registry keys (HKCU\Software\Classes\mscfile and .pwn) that reference shellcode (likely a downloaded component) or its own path. When Windows attempts to run the legitimate tool, the registry keys redirect execution, launching the malware with elevated privileges.
- Fodhelper.exe Bypass: It associates the ms-settings ProgID with the custom .pwn extension in HKCU\Software\Classes\ms-settings\CurVer and sets the file path of its own sample in HKCU\Software\Classes\.pwn\Shell\Open\Command. This key chain allows the malware to execute automatically when the legitimate Fodhelper.exe is triggered.

2. Access Token Manipulation (T1134)

To ensure full control, ValleyRAT adjusts its security token to enable the SeDebugPrivilege. This privilege allows the malware to interact with, inspect, and even terminate processes belonging to other users or higher integrity levels. It also uses the internal NTdll.dll function NtSetInformationProcess with class 29, a known anti-debugging maneuver.

```
// Adjusting token privileges to enable SeDebugPrivilege
LookupPrivilegeValue(0, L"SeDebugPrivilege", (PLUID)NewState.Privileges);
NewState.Privileges[0].Attributes = 2; // SE_PRIVILEGE_ENABLED
```

```
AdjustTokenPrivileges(TokenHandle, 0, &NewState, 0x10u, 0, 0);
// Anti-debugging through native API
result = (int)GetProcAddress(v1, "NtSetInformationProcess");
```

Anti-Defense & Evasion Strategies

ValleyRAT's extensive evasion repertoire is notable, targeting everything from virtualization to specific anti-virus products.

1. Targeting Security Products (T1562.001)

The malware contains an exhaustive hardcoded list of executable names belonging to popular anti-virus products and Host-based Intrusion Prevention Systems (HIPS), predominantly from Chinese vendors like Qihoo 360 and Tencent QQ PC Manager. Its logic is to terminate these processes (leveraging its new SeDebugPrivilege) before proceeding. It also modifies security software's registry settings to disable their autostart capability upon reboot.

Vendor (Examples)	Targeted Executables (Examples)
Qihoo 360	360d.exe, 360Safe.exe, 360Tray.exe
Tencent QQ	QQPCRTP.exe, QQMPersonalCenter.exe
Kingsoft	kxscan.exe, kwsprt.exe, kxascore.exe

2. Windows Defender Bypass via PowerShell

To neutralize Microsoft Defender, the malware programmatically launches a shell and injects a PowerShell command to exclude itself from real-time monitoring.

```
// PowerShell command injected to bypass Windows Defender
mw_setup_remote_shell(
  Buffer,
  "Invoke-Command -Command {Add-MpPreference -ExclusionPath \"%s:\\\\\"}\"\\r\\n",
  v16->m128i_i8);
// Debug string confirms the target
OutputDebugStringA("RC|WinDefend");
```

The command Add-MpPreference -ExclusionPath is used to effectively exclude the entire drive path where the malware resides from future Defender scans, achieving effective defense evasion.

3. Anti-Analysis and Sandbox Evasion (T1497)

ValleyRAT is highly suspicious of its environment, performing checks that fall into two main categories:

Technique	Description
Process/Window Hunting	It enumerates running windows and checks their title strings for known analysis tools such as Wireshark, Fiddler, Malwarebytes, ApateDNS, and TaskExplorer.
Anti-Virtualization	It uses the **CPUID** instruction to check for the **"GenuineIntel"** or **"AuthenticAMD"** vendor strings, which are often spoofed or missing in virtual environments like VMware or VirtualBox. It also checks for the existence of VMware directories and limited memory conditions.

Persistence and C2 Readiness

After clearing the defenses, ValleyRAT secures its foothold and prepares for communication.

1. Persistence via Registry Run Key (T1547.001)

A standard, but effective, persistence mechanism is used by writing its own execution path to the HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run key using a deceptive value name, GFIRestart32.exe. This guarantees execution on every user login.

2. Command and Control (C2) Beacon (T1071)

Before attempting to contact its actual C2 server (whose IP/Port are stored in obfuscated registry entries like HKCU\Software\Console\IpDate), the malware performs an initial Internet Connection Discovery check.

```
// Checking Internet connectivity via a Chinese domain
while ( !InternetCheckConnectionA("http://www.baidu.com", 1u, 0) )
// ... payload generation ...
mw_setup_remote_shell(Buffer, "GetOnlineSize_%d", v3);
return sub_1B0002D70(Buffer);
```

By querying hxxp://www[.]baidu.com, it ensures the network is alive. It then generates a randomized integer (v3) to construct a dynamic beacon string (e.g., GetOnlineSize_12345) which is sent to the C2 server, likely to download the next-stage payload.

What Features Differ ValleyRAT Malware from Other Families?

ValleyRAT distinguishes itself from many other malware families through its highly targeted nature and aggressive, multi-layered approach to evasion and privilege escalation. It begins with a unique geographical kill switch, checking for specific Chinese application registry keys (WeChat, DingTalk) before proceeding, ensuring it only executes in its intended operational environment.

Once active, it aggressively pursues system mastery by using **multiple UAC bypasses** (exploiting Fodhelper.exe, Event Viewer, etc.) and seizing the SeDebugPrivilege on its process. For stealth, it favors loading its payload into the legitimate system binary MSBuild.exe and actively disables defenses by:

- Terminating known AV processes (especially Chinese vendors).
- Abusing native PowerShell cmdlets to add its paths to the Windows Defender exclusion list.
- Using**CPUID**and window title checks for robust anti-analysis and anti-virtualization.

This combination of specialized targeting, simultaneous UAC attack vectors, and systematic defense impairment makes ValleyRAT a particularly sophisticated and resilient threat.

How Picus Helps Defend Against ValleyRAT Malware Attacks?

The Picus Security Validation Platform safely simulates ValleyRAT malware attacks, replicating the malware's new plug-in functionality for validation purposes. Through the Picus Threat Library, it replicates the tactics, techniques, and procedures (TTPs) observed in these campaigns to reveal detection and prevention gaps across EDR, NGFW, and SIEMtechnologies, before adversaries can exploit them.

You can also test your defenses against hundreds of other malware variants, such as SnipBot, SlipScreen Loader, RustyClaw, within minutes with a 14-day free trial of the Picus Platform.

Threat ID	Threat Name	Attack Module
29426	ValleyRAT Malware Downloader Download Threat	Network Infiltration
25204	ValleyRAT Malware Downloader Email Threat	Email Infiltration
59821	ValleyRAT Loader Download Threat	Network Infiltration
54856	ValleyRAT Loader Email Threat	Email Infiltration

72873	ValleyRAT Malware Dropper Download	Network Infiltration
46588	ValleyRAT Malware Dropper Email Threat	Email Infiltration

Key Takeaways

- Multi-Stage Execution: ValleyRAT utilizes a multi-stage loader (often .NET-based) that decrypts and loads components entirely in memory.
- LOLBin for Stealth: The final payload is typically injected into the legitimate Windows binary MSBuild.exe (a Living Off the Land Binary) to masquerade as a trusted process.
- Targeted Kill Switch: It contains a geographically specific kill switch that checks the registry for Chinese communication apps (WeChat and DingTalk). If not found, the malware terminates to prevent analysis.
- Multiple UAC Bypasses: It employs three distinct techniques for Privilege Escalation (UAC Bypass), exploiting vulnerabilities in Windows components like Fodhelper.exe, Event Viewer, and CompMgmtLauncher.exe.
- Token Manipulation: It manually enables the SeDebugPrivilege on its security token, granting it power to inspect and terminate other processes.
- Aggressive Anti-AV: It carries an extensive list of known security program executables (from Qihoo 360, Tencent, etc.) to actively terminate or disable their autostart capabilities.
- Windows Defender Exclusion: It uses a programmatic PowerShell command **(Add-MpPreference -ExclusionPath)** to add its files or entire drives to the Windows Defender exclusion list.
- Anti-Analysis (VM/Sandbox): It performs environmental checks, including the CPUID instruction to verify the CPU vendor ID (checking for "GenuineIntel" or "AuthenticAMD") and enumerating window titles to detect analyst tools (Wireshark, Fiddler, etc.).
- Persistence: It establishes persistent execution through both Registry Run Keys (using a deceptive name like GFIRestart32.exe) and by copying itself to the Startup folder.
- Dynamic C2 Beacon: It performs an Internet check using hxxp://www[.]baidu.com and generates a randomized integer to construct a unique C2 beacon string, aiding in network evasion.