The “Hikit” Rootkit: Advanced and Persistent Attack Techniques (Part 1) « The “Hikit” Rootkit: Advanced and Persistent Attack Techniques (Part 1)

By by Ryan Kazanciyan

Published: 2012-08-20 · Archived: 2026-04-05 17:21:49 UTC

By Ryan Kazanciyan and Christopher Glyer

Mandiant's analysts routinely identify new backdoors, rootkits, and other custom-developed malware during the course of our investigations - especially those related to targeted attacks performed by sophisticated actors. Although much of this malware is relatively mundane (or derived from publicly available tools), we occasionally encounter a particularly interesting sample. The "Hikit" rootkit is one such example, combining advanced capabilities with exceptionally clever mechanisms for persistence and hiding on a host.

We first encountered this malware during a sweep of thousands of systems in a victim environment for Indicators of Compromise (IOCs), using our Mandiant Intelligent Response (MIR) platform. The attacker already had administrator privileges to the entire corporate Windows domain and had compromised numerous systems. Fortunately, we had several indicators gathered during the onset of the investigation that we could use during initial MIR sweeps. For instance, we knew they were fond of using the old-but-reliable"sticky keys" technique, whereby "sethc.exe" is overwritten with a copy of "cmd.exe"to provide unauthenticated access during RDP logon. (Carnal0wnage's blog has a nice succinct write-up of this attack here.)

Sure enough, our sweeps discovered several hosts with evidence of tampered "sethc.exe". We collected Live Response data (file listing, registry listing, event logs, URL history, prefetch, restore point analysis, memory , etc.) from each system that returned an IOC hit in order to facilitate more in-depth analysis.

We used the file system time metadata from the sweep results as a starting point to help us build a timeline of attacker activity across all collected evidence. On one system (a Windows 2003 server) this process identified two suspicious files that were created in succession within the same hour as the sticky-keys attack:

- C:WINDOWSsystem32wbemoci.dll
- C:WINDOWSsystem32driversW7fw.sys

We provided the samples to our malware analysis team, and their initial triage results indicated that "oci.dll" was a dropper and loader for rootkit driver code in "W7fw.sys". But how was "oci.dll" being loaded at installation or boot time? There was no evidence of rundll32.exe being executed to manually call one of the DLL's exported functions. There were no registry references to the file, nor was it being loaded through search order hijacking. After reviewing several hosts infected with the same malware, we had our answer.

On each infected host, the System event log recorded the same sequence of events immediately following the creation of "oci.dll":

- "The MS DTC service is stopping", followed by
- "The Distributed Transaction Coordinator service was successfully sent a start control", followed by
- Creation of "C:WINDOWSsystem32driversW7fw.sys"at the same second

"MS DTC" is the abbreviated name for the Microsoft Distributed Transaction Coordinator service, a legitimate Windows component. We checked the ImagePath binary for this service, "msdtc.exe", and confirmed that it was digitally signed and verified by Microsoft and had an MD5 matching known-good versions of the file.

Next, we did some research on how this service works and on the DLL dependencies that it imports in order to run. We discovered that msdtc.exe binary has an import "msdtctm.dll" which loads"mtxoci.dll" (supports the Microsoft ODBC Driver for Oracle), which in turn loads "oci.dll" - which can be a legitimate Oracle-related library. Microsoft even had a knowledge base article on how "msdtc.exe" can cause "oci.dll" to start on certain versions of Windows: http://support.microsoft.com/kb/233297. Talk about an obscure persistence mechanism - the bad guys certainly did their research!

A few weeks later we encountered another infected system that had remnants of a batch script "1.bat" used by the attacker during the rootkit installation process:

```
sc config msdtc obj= localsystem pass-
word= ""
sc failure msdtc reset= 0 actions=
restart/5000
sc config msdtc start= auto
sc config msdtc type= own type= inter-
act
net stop msdtc /y
net start msdtc
```

This validated our initial theory: the attacker used the script to configure, stop, and re-start the "msdtc" service after dropping "oci.dll" on the system.

Since discovering this persistence mechanism, we have identified multiple targeted threat groups using the same technique to load different malware.

In the second part of this blog post, we'll provide more detail on the rootkit's functionality and counter-forensic techniques and put together an Indicator of Compromise (IOC) that embodies all of its characteristics. We'll also provide insight into how the attacker took advantage of the rootkit's functionality within the context of a large-scale intrusion.

Source: https://web.archive.org/web/20190216180458/https://www.fireeye.com/blog/threat-research/2012/08/hikit-rootkit-advanced-persistent-attack-techniques-part-1.html