Raspberry Robin gets the worm early

By Anna Seitz

Archived: 2026-04-05 13:14:54 UTC

Initial access

Raspberry Robin is typically introduced via infected removable drives, often USB devices. The Raspberry Robin worm often appears as a shortcut `.lnk` file masquerading as a legitimate folder on the infected USB device.

Soon after the Raspberry Robin infected drive is connected to the system, the UserAssist registry entry is updated and records execution of a ROT13-ciphered value referencing a `.lnk` file when deciphered. In the example below, `q:\erpbirel.yax` deciphers to `d:\recovery.lnk`.

Figure 2: Registry modification with ROT13 `.lnk` file

Execution

Raspberry Robin first uses `cmd.exe` to read and execute a file stored on the infected external drive. The command is consistent across Raspberry Robin detections we have seen so far, making it reliable early evidence of potential Raspberry Robin activity. Typically the command line includes `cmd /R <` to read and execute a file. The use of `cmd /R <` is not unique to Raspberry Robin, but the filename pattern is unique. The filename is made up of five to seven random alphanumeric characters and a variety of file extensions. Some of the file extensions we've seen include `.usb`, `ico`, `.lnk`, `.bin`, . `sv`, and `.lo`. Additionally, the command has sometimes included type, which is a built-in command to display the contents of a file.

Here's an example of what the whole command might look like:

Figure 3: Raspberry Robin `cmd.exe` command

Next, `cmd.exe` typically launches `explorer.exe` and `msiexec.exe`. With Raspberry Robin, `explorer.exe`'s command line can be a mixed-case reference to an external device; a person's name, like `LAUREN V`; or the name of the `.lnk` file, like the figure below. The name here has been modified from the `.lnk` file name to `LNkFILe`. While we aren't sure of this command's exact purpose, we've consistently observed it in Raspberry Robin detections.

Figure 4: Mixed-case command referring to device or name

Raspberry Robin extensively uses mixed-case letters in its commands. Adversaries sometimes use mixed-case syntax in an attempt to evade detection. Case-sensitive, string-based detections written to detect `evil` may not fire on `eViL`, but `cmd.exe` is case-insensitive and has the flexibility to read and process both commands the same way.

Command and control (C2)

Let’s look at Raspberry Robin’s `msiexec.exe` command in detail, since that informs our first behavior-based detection opportunity.

While `msiexec.exe` downloads and executes legitimate installer packages, adversaries also leverage it to deliver malware. Raspberry Robin uses `msiexec.exe` to attempt external network communication to a malicious domain for C2 purposes. The command line has several key features we have seen across multiple detections:

- Use of mixed-case syntax (this is yet another example of mixed case use by Raspberry Robin)
- Use of short, recently-registered domains only containing a few characters, for example `v0[.]cx`
- The domains in our detections hosted QNAP NAS device login pages around the time of the Raspberry Robin activity. We hypothesize Raspberry Robin may use compromised QNAP devices for C2 infrastructure. The use of (ostensibly) compromised QNAP devices for C2 infrastructure is not unique to this activity cluster, but we observed operators using these across several Raspberry Robin-associated detections.
- Inclusion of port `8080`, a non-standard HTTP web service port, in the URL
- Inclusion of a string of random alphanumeric characters as the URL subdirectory, frequently followed by the victim’s hostname and username

Here is a modified example of a full malicious Raspberry Robin `msiexec.exe` command line matching all of the above criteria. The random string has been modified, and the victim’s host name replaced with `HOSTNAME`, though the domain name remains the original one observed.

Figure 5: Malicious Raspberry Robin `msiexec.exe` command

To detect suspicious use of `msiexec.exe` by Raspberry Robin or other threats, it’s essential to take a look at the command line and the URL. Detecting `msiexec.exe` making outbound network connections to download and install packages in the command line interface will give you the opportunity to examine the activity and determine if it’s malicious or not.

Detection opportunity: `msiexec.exe` downloading and executing packages

Identify the use of Windows Installer Tool `msiexec.exe` to download and execute packages in the CLI.

process == `('msiexec')`
&&
process_command_line_includes == `('http:', 'https:')`

&&
process_command_line_includes == `('/q', '-q')`

Persistence

In several Raspberry Robin detections, we have seen `msiexec.exe` go on to install a malicious DLL file. At this time we are not certain what the DLL does.. We suspect it may establish persistence on the victim's system. In the detections we saw, the malicious files were created as `C:\Windows\Installer\MSI****.tmp` files. In one case, a file with the same hash was also created as `C:\Users\username\AppData\Local\Temp\bznwi.ku`.

Examples:

- `C:\Windows\Installer\MSI5C01.tmp`
  `C:\Users\username\AppData\Local\Temp\bznwi.ku`
  - Shared MD5 hash: 6f5ea8383bc3bd07668a7d24fe9b0828
  - VirusTotal example
- `C:\Windows\Installer\MSIE160.tmp`
  - MD5 hash: e8f0d33109448f877a0e532b1a27131a
  - VirusTotal example

Execution (again)

Next, `msiexec.exe` launches a legitimate Windows utility, `fodhelper.exe`, which in turn spawns `rundll32.exe` to execute a malicious command. Processes launched by `fodhelper.exe` run with elevated administrative privileges without requiring a User Account Control prompt. It is unusual for `fodhelper.exe` to spawn any processes as the parent, making this another useful detection opportunity.

Detection opportunity: `fodhelper.exe` as a parent process

Identify Windows Features On Demand helper `fodhelper.exe` creating processes as the parent.

parent_process == `('fodhelper')`

The `rundll32.exe` command starts another legitimate Windows utility, in this case `odbcconf.exe`, and passes in additional commands to execute and configure the recently-installed malicious DLL `bznwi.ku` (Hash: `6f5ea8383bc3bd07668a7d24fe9b0828`). Here is what that command looks like. (We modified the random string values in the command, as well as replaced the victim's username with `username`.)

Figure 6: Malicious `rundll32.exe` command

The `-A` flag in `odbcconf.exe` specifies an action. `configdriver` loads the driver setup DLL, in this case `VKIPDSE`. `SETFILEDSNDIR` creates the registry location `HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI\ODBC File DSN\DefaultDSNDir`, if it does not already exist, and specifies the default location used by the ODBC Data Source Administrator when creating a file-based data source. `INSTALLDRIVER` adds additional information about the driver.

In this detection, we saw `odbcconf.exe` successfully execute the malicious command. Since `odbcconf.exe` has a built-in `regsvr` flag similar to `regsvr32.exe`, it can be used by adversaries to execute DLLs and bypass application control defenses that aren't monitoring for `odbcconf.exe` misuse.

Detection opportunity: `odbcconf.exe` loading .DLLs

Detect the Windows Open Database Connectivity utility loading a configuration file or DLL. The /A flag specifies an action, `/F` uses a response file, and `/S` runs in silent mode. `Odbcconf.exe` running rgsvr actions in silent mode could indicate misuse.

process == `('odbcconf')`
&&
process_command_line_includes == `('regsvr)`
&&
process_command_line_includes == `('/f', '-f')`
||
process_command_line_includes == `('/a', '-a')`
||
process_command_line_includes == `('/s', '-s')`

C2, part deux

We observed outbound C2 activity involving the processes `regsvr32.exe`, `rundll32.exe`, and `dllhost.exe` executing without any command-line parameters and making external network connections to IP addresses associated with TOR nodes. Additionally, some of the IP addresses in the connections host domains consisting of random alphanumeric characters. For example, `hxxps[:]//www[.]ivuoq6si2a[.]com/`.

This activity presents us with a final detection opportunity. It is atypical for `regsvr32.exe`, `rundll32.exe` and `dllhost.exe` to execute with no command-line parameters and establish external network connections. This behavior is not inherently malicious, but is good to monitor.

Detection opportunity: network connections from the command line with no parameters

Detect `regsvr32.exe`, `rundll32.exe`, and `dllhost.exe` making external network connections with an empty command line.

process == `('regsvr32')`
||
process == `('rundll32')`
||
process == `('dllhost')`
&&
process_command_line_contains == `("")`
&&
has_netconnection

**Note: Double Quotes ("") within the command line means null.*

Source: https://redcanary.com/blog/raspberry-robin/