Quasar Open-Source Remote Administration Tool | CISA

Published: 2019-02-14 · Archived: 2026-04-05 22:59:23 UTC

Quasar is a publically available, open-source RAT for Microsoft Windows operating systems (OSs) written in the C# programming language. Quasar is authored by GitHub user MaxXor and publicly hosted as a GitHub repository. While the tool can be used for legitimate purposes (e.g., an organization's helpdesk technician remotely accessing an employee's laptop), the Cybersecurity and Infrastructure Security Agency (CISA), is aware of APT actors using Quasar for cybercrime and cyber espionage campaigns.

Quasar was first released in July 2014 as "xRAT 2.0." In August 2015, xRAT was renamed "Quasar" and released as v1.0.0.0. For this report, the National Cybersecurity and Communications Integration Center (NCCIC), part of CISA, analyzed Quasar version 1.3.0.0, which was released on September 28, 2016, and is the latest stable version available on GitHub. This report does not reflect any changes Quasar's author has made to the tool's source code since the release of v1.3.0.0.

Open-source reports state that some APT actors have adapted Quasar and created modified minor (1.3.4.0) and major (2.0.0.0 and 2.0.0.1) versions.[1] ,[2] NCCIC has not determined the exact difference between these versions and v1.3.0.0. Therefore, NCCIC cannot definitively say whether the detection and mitigation recommendations provided in this report will work effectively against APT actor-modified versions of Quasar.

High Level Architecture

Quasar uses a client-server architecture that enables one user to remotely access many clients. The server is responsible for creating client binaries and managing client connections. Users then interact with connected clients through the server's graphical user interface (GUI).

Note: Quasar does not contain software vulnerability exploits. Threat actors must leverage other tools or methods to gain access to a target host before they can use Quasar.

Requirements

Quasar requires a Microsoft .NET Framework 4.0 (or higher) Client Profile. The Quasar client and server will run on the following OSs (32- and 64-bit):

- Windows XP Service Pack 3,
- Windows Server 2003,
- Windows Vista,
- Windows Server 2008,
- Windows 7,
- Windows Server 2012,
- Windows 8/8.1, and
- Windows 10.

Quasar Server

The Quasar server component is responsible for

- Listening for and handling client connections (e.g., catching new connections, terminating connections);
- Managing connected clients (e.g., retrieving files, showing the screen, killing processes); and
- Configuring and building client executables.

Figure 1 shows the Quasar server component GUI. Quasar users interact with the server and, in turn, its clients, through the GUI. Each client's entry is listed individually and includes the client's Internet Protocol (IP) address, username, Quasar client version, connection status, user status, country, OS, and account type. The Quasar user initiates client interactions by right-clicking an individual client row, which opens a pop-up menu with available commands.

Figure 1: Quasar screenshot – example of a Quasar server with a connected client

The server component builds client executables that the Quasar user can run on target hosts. The client builder feature allows the Quasar user to select from different options and attributes (see table 1).

Table 1: Quasar client builder feature options and attributes

Option	Default Option	Description
Basic Settings		
Client tag	None	Represents the name for the client instance. This value is displayed in the connection table (see figure 1) of the Quasar server GUI once the client connects
Mutex	QSR_MUTEX_[18 character alphanumeric upper and lowercase string]	Sets the file mutual exclusion object (mutex) to prevent the same host being infected multiple times
Connection Settings		
Callback IP	None	Sets the server IP for the client connection
Callback domain	None	Sets the domain for the client connection
Callback port	4782	Sets the Transmission Control Protocol (TCP) port callback to “on”
Password	1234	Sets the password for Advanced Encryption Standard (AES) encryption
Connection retry	300ms	Sets how often the client will attempt to callback if they are not connected
Installation Settings		
Install client	Off	Sets the default for whether or not the client will install on a host
Base installation paths	• `%AppData%` • Program Files* • `Windows\SysWOW64` *	The location where the client file will be installed on a host. This field is limited to the options listed. Starred items (*) require administrator privileges
Install subdirectory	SubDir	Makes a customizable subdirectory within the base installation path
Install name	Client	The name of the client file. This file must be `.exe`
Run client when the computer starts	Off	A checkbox that, if checked, will add the Quasar client as an AutoRun via Registry Key or Scheduled Task

Startup name	Quasar Client Startup	Customizable free text field

The Quasar user can also set metadata to be embedded in the executable, such as the author, organization, copyright, year, and version.

Quasar Client

Quasar client instances are built by the server component. Based on multiple client builds, each with different configurations, the client size is consistently 349KB. Once it is distributed to a target host, the client needs to be executed before it can call back to the server. Client execution is invisible to the target host user and does not generate any visible windows or notifications on the target host, except in cases where the client becomes unresponsive. Once running on a target host, the client process is visible to the target host user via Windows Task Manager or a similar process management program.

Network Traffic

Quasar encrypts communications using the AES algorithm. The client builder hardcodes a Quasar user-chosen, pre-shared key to be used in command and control (C2) communications. The server must be configured to listen on the callback port and use the pre-shared key. (Quasar's author has stated [via GitHub] that they would like to update Quasar to use Transport Layer Security for C2 encryption in the future.)

After the TCP handshake is completed, all traffic between the server and client is encrypted. The entropy of AES ciphertext makes it impossible to write a pattern to detect this content. Quasar uses the first 4 bytes of the TCP payload to track the payload's total size in little-endian format. This size-tracking pattern is distinctive to Quasar network traffic. As shown in figure 2, the first 4 bytes of the TCP payload contain 0x40000000 or 64 decimal in hexadecimal notation. Subtracting the tracking bytes (4 bytes) from the total TCP payload (68 bytes) results in an actual payload size of 64 bytes.

Figure 2: Quasar screenshot – C2 traffic

The distinctive first 4 bytes of the payload can be used to identify Quasar traffic. Specifically, the first 4 bytes can identify the first packet sent from the server to the client following the TCP handshake. This packet is used to initiate the server/client authentication process. See table 2 for a description of the attributes of the first packet from the server to the client following the TCP handshake. This information can be used to identify potential Quasar activity on a network.

Table 2: Quasar packet attributes

Type	Attribute
TCP Flag	`PSH` and `ACK`
Total Size	122 bytes
TCP payload size	68 bytes
First 4 bytes of payload	0x40000000

Client Network Traffic

Quasar allows the user to gather host system information. As part of the client connection setup, the client attempts to discover its geolocation—including its Wide Area Network (WAN) IP address—by sending an `HTTP GET` request to the Uniform Resource Locator (URL) `ip-api[.]com/json/` with User-Agent string:

`Mozilla/5.0 (Windows NT 6.3; rv:48.0) Gecko/20100101 Firefox/48.0`.

This User-Agent string mimics a Mozilla Firefox 48 browser running on Windows 8.1. This User-Agent string would likely stand out as unique in a corporate network environment, and its presence could be a high-confidence indication of Quasar activity.

If the client does not receive a response from this lookup, the client attempts to retrieve WAN IP information from `freegeoip[.]net` and `api[.]ipify[.]org`, respectively. The User-Agent string remains consistent across all attempts.

Quasar users can also direct the client to access websites. These requests can be set as visible to the host user via a browser window that opens or invisible to the host user via the C# WebRequest class. Requests that are visible to the host user use the User Agent string from the Quasar user's browser. Requests that are marked as invisible to the host user are sent with User-Agent string:

`Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.75.14 (KHTML, like Gecko) Version/7.0.3 Safari/7046A194A`.

This User-Agent string mimics an Apple Safari 7.0.3 browser running on Mac OS X 10.9.3. The use of Mac OS X as the operating system is interesting because Quasar can only be run on Windows. NCCIC has leveraged Quasar's use of Mac OS X to limit false positives in the Snort signatures for this activity.

The User-Agent strings listed in this section are set by the server component when the client file is built. The strings can only be changed by altering the User-Agent string in the server source code. All clients built with a server component compiled from unaltered Quasar v1.3.0.0 source code contain these User-Agents.

Client Installation

Quasar's client builder limits the base directories in which the client may be placed. The three base directories in which the Quasar client builder can place itself are

- Program Files (requires administrator privileges),
- `Windows\SysWOW64` (requires administrator privileges), and
- %APPDATA%.

Figure 3: Quasar sceenshot – client installation settings

Quasar users can specify which subdirectory within the base directory to place the client executable (as shown in figure 3). Quasar users can also specify the name of the executable. Both the client executable and the subdirectory can be hidden from the target host user during installation by a Windows application programming interface call that sets one of the file's attributes to "hidden." The "hidden" setting only hides files from the target host user's view in Windows File Explorer.

Persistence

Quasar achieves persistence by executing on startup, as seen in the source code shown in figure 4. To achieve persistence, Quasar uses two methods: scheduled tasks and registry keys. If the client process has administrator privileges, the client will generate a scheduled task via `schtasks`. The scheduled task is generated using the task name created in the client builder. The schedule task runs after the host user logs on, executes with the highest run level (i.e., the highest level of privilege), and suppresses any errors related to creating the task. If the process does not have administrator privileges, the scheduled task will only add a registry value. That registry value is added to the following key:

`HKCU\Software\Microsoft\Windows\CurrentVersion\Run`.

The value name is then configured in the client builder, and the client adds its current path as the startup program.

```
if (WindowsAccountHelper.GetAccountType() == "Admin")
        {
           try
           {
              ProcessStartInfo startInfo = new ProcessStartInfo("schtasks")
              {
                 Arguments = "/create /tn \"" + Settings.STARTUPKEY + "\" /sc ONLOGON /tr \"" +
ClientData.CurrentPath + "\" /rl HIGHEST /f",
                 UseShellExecute = false,
                 CreateNoWindow = true
              };
              Process p = Process.Start(startInfo);
```

```
            p.WaitForExit(1000);
            if (p.ExitCode == 0) return true;
        }
        catch (Exception)
        {
        }
        return RegistryKeyHelper.AddRegistryKeyValue(RegistryHive.CurrentUser,
            "Software\\Microsoft\\Windows\\CurrentVersion\\Run", Settings.STARTUPKEY,
ClientData.CurrentPath,
            true);
    }
    else
    {
        return RegistryKeyHelper.AddRegistryKeyValue(RegistryHive.CurrentUser,
            "Software\\Microsoft\\Windows\\CurrentVersion\\Run", Settings.STARTUPKEY,
ClientData.CurrentPath,
            true);
```

Figure 4: Source code from `Quasar/Client/Core/Installation/Startup.cs`

Privilege Escalation

Quasar allows the tool user to escalate the client's running privileges, as seen in the source code shown in figure 5. To escalate the client's running privileges, Quasar attempts to launch a command prompt (`cmd.exe`) as an administrator. The elevated command prompt then relaunches the client. The client inherits the parent process' now-elevated privileges. If the Window's User Account Control (UAC) is configured, this method generates a UAC pop-up window on the target host, which asks the target host user to confirm the process of running the command prompt as the administrator.

```
if (WindowsAccountHelper.GetAccountType() != "Admin")
    {
        ProcessStartInfo processStartInfo = new ProcessStartInfo
        {
```

```
            FileName = "cmd",
            Verb = "runas",
            Arguments = "/k START \"\" \"" + ClientData.CurrentPath + "\" & EXIT",
            WindowStyle = ProcessWindowStyle.Hidden,
            UseShellExecute = true
};
```

Figure 5: Source code from `Quasar/Client/Core/Commands/SystemHandler.cs`

Summary

Quasar, a legitimate open-source remote administration tool (RAT), has been observed being used maliciously by Advanced Persistent Threat (APT) actors to facilitate network exploitation.

This Analysis Report provides information on Quasar's functions and features, along with recommendations for preventing and mitigating Quasar activity.

Solution

Network defenders can detect Quasar activity by monitoring network traffic for its unique pattern, the registry key it edits for persistence, mutexes for strings that follow the default Quasar pattern, and the directories where Quasar installs itself. Commercial antivirus programs detect most Quasar client binary builds as malicious.

Snort Signatures

Signature 1: TCP Payload Size Tracking

This signature matches on a server-to-client packet with a TCP payload length of 68 bytes and the first 4 bytes matching the size tracking sequence. NCCIC observed this packet as the first packet after the TCP handshake. Network defenders can create and implement additional signatures to detect differing TCP payload sizes and the packet's respective size tracking sequences. The following Snort signature can be used to detect unmodified Quasar v1.3.0.0; however, it is unknown if this signature can be used to detect modified versions.

alert tcp $EXTERNAL_NET :1024 -> $HOME_NET any (msg:"Non-Std TCP Server Traffic contains '|40 00 00 00|' (Quasar RAT Initial Packet)"; sid:10000; rev:1; flow:established,from_server; dsize:68; content:"|40 00 00 00|"; depth:4; fast_pattern;)

Quasar uses a TCP payload of 68 bytes at the beginning of each of its sessions. Quasar's distinctive 68-byte TCP payload presents the best opportunity for network defenders to identify Quasar activity.

It is likely that the Quasar TCP payload server packet will originate from TCP port 80 or 443 to traverse network firewalls and attempt to blend in with normal web browsing traffic. Network defenders may want to further limit this Snort signature to only TCP ports 80 or 443.

When reviewing alerts generated by this Snort signature, network defenders should look for server-to-client TCP `PSH/ACK` packets following the alert packet. True positive alerts will likely have a 4-byte tracking sequence equal to the size of the TCP payload minus 4 bytes, with what appears to be ciphertext in the remaining payload.

NCCIC recommends applying this Snort signature to a network sensor located on an organization's perimeter to limit the false positives generated by internal organization traffic.

Signature 2: IP Lookup User-Agent String, HyperText Transfer Protocol Header Host, and HyperText Transfer Protocol Header URI

alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"HTTP Client Header contains 'Host|3a 20|ip-api com', URI '/json/' (Quasar RAT)"; sid:10002; rev:1; flow:established,to_server; content:"Host|3a 20|ip-api|2e|com|0d 0a|"; http_header; fast_pattern:only; content:"User-Agent|3a 20|Mozilla/5.0 (Windows NT 6.3|3b| rv|3a|48.0) Gecko/20100101 Firefox/48.0|0d 0a|"; http_header; content:"/json/"; http_uri; depth:6; urilen:6,norm; priority:2;)

This Snort signature alerts on the WAN IP lookup initiated by the Quasar client. The User-Agent string, Hypertext Transfer Protocol (HTTP) header host, and HTTP header URI values are set by the server component when the client is built. The server component is configured with these values at compile time. The User-Agent string mimics Windows 8.1 running Firefox 48, both of which are considerably dated. It is possible to see this User-Agent string used legitimately; however, organizations with information technology baselines should know if this User-Agent string legitimately exists in their network environment.

Signature 3: Hidden HTTP Request User-Agent String and Time-to-Live

alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"HTTP Client Header contains 'User-Agent|3a 20|Mozilla/5.0 (Macintosh|3b| Intel Mac OS X 10_9_3) AppleWebKit/537.75.14 (KHTML, like Gecko) Version/7.0.3 Safari/7046A194A', TTL 65-128 (Quasar RAT)"; sid:10001; rev:1; flow:established,to_server; content:"User-Agent|3a 20|Mozilla/5.0 (Macintosh|3b| Intel Mac OS X 10_9_3) AppleWebKit/537.75.14 (KHTML, like Gecko) Version/7.0.3 Safari/7046A194A|0d 0a|"; http_header; fast_pattern:only; priority:2;)

This Snort signature alerts on a client-generated hidden HTTP request. The Quasar user can direct the target host to visit a URL and retrieve the content. If the request is set to "hidden," the client uses this User-Agent string to mimic Mac OS X 10.9.3 and Safari 7. Mac OS X 10.9.3 and Safari 7 are not only dated, but also do not match the OS on which Quasar operates (i.e., Windows). When reviewing alerts NCCIC recommends looking for packets with a TTL between 65 and 128.

References

FireEye blog on new tools used by an APT group

Palo Alto Networks Unit 42 blog on Quasar

Revisions

December 18, 2018: Initial version|December 21, 2018: Updated signatures

---