QRadar Ariel Data Export

Date: November 2025

Paul Vinson - dl-qradar-data-export@paloaltonetworks.com

Table of Contents

QRadar Ariel Export Process Overview

This document provides a detailed overview of the QRadar Ariel Export process for QRadar captured events, which involves extracting, transforming, and exporting QRadar Ariel records into a usable format, typically gzipped JSON files. The process leverages a combination of Bash, Perl, and Java utilities, with key components working in conjunction to ensure efficient and scalable data handling. Any required Perl or Java components are installed and provided on a QRadar host, and this process will only function on a correctly-configured QRadar host. The only external dependency is the GNU Parallel package from https://www.gnu.org/software/parallel/, which is provided for you as part of this package.

Note: This extraction/conversion process is only for events, not flows. Flow record extraction/conversion is not supported at this time.

Preparation:

QRadar on-prem:

1) This set of utilities can be run on any QRadar host (virtual or physical) that has Ariel data that needs to be extracted/converted. Not recommended to be run on a production QRadar host ***unless ingestion is quiesced***. In fact, only the main PostgreSQL database needs to be running and available, all other QRadar processes can and should be stopped as the QRadar auto-tuning processes will start shutting down production processes as loads are incurred from this package. As part of this package there is a script provided that can shutdown all unnecessary QRadar processes "stop_all_qradar.sh".
2) Ensure that there is sufficient space for resulting output files in compressed JSON format.
3) This QRadar host can be run in a virtual environment with large resources allocated (vCPU intensive) to take advantage of massively parallel operations.

QRadar on Cloud (QRoC):

1) QRoC customer submits ticket to IBM SRE (Devops) and provides authenticated cloud storage of some type along with access details, and requests A) uncompressed copy of their Ariel Event Database files from all Ariel nodes (records and payloads, and checksum files if generated and available) and B) copy of their QRadar configuration backup and C) output of below referenced script "`extract_psql_device_info.pl`" which will be two files, "`log_source_mappings.out`" and "`qid_mappings.out`". IBM SRE will have possession of the script "`extract_psql_device_info.pl`". The customer simply needs to request the output files as part of the ticket request.

2) IBM SRE will copy all requested Ariel DB files to customer-provided storage. This request is normally satisfied fairly quickly as it is a straightforward copy of data; no extraction/conversion is required on their part. Note that Ariel DB files from multiple Ariel nodes will be munged together; this is fine as the filenames are collision-free and will happily reside together and can be searched through normal Ariel queries (later, if appropriate) and correctly exported/converted via this package.
3) The customer will either install a minimally-EPS-licensed version of QRadar or QRadar Community Edition (CE) and mount the cloud storage locally where the Ariel DB records reside. This image can be run in a virtual environment with large resources allocated (vCPU intensive) to take advantage of massively parallel operations.
4) The customer can optionally restore the configuration backup supplied by IBM SRE and go through the process of removing all distributed QRadar hosts, effectively turning the deployment into an All-in-One (Console AIO) configuration. Besides providing the ability to extract/convert Ariel records, this option also enables the customer to query the Ariel database through the GUI as normal, using all customer-derived building blocks and customized rule content and custom extracted properties (CEP), if any. Customers should not expect these queries to perform like a production distributed environment, due to the nature of the much-smaller deployment (Console AIO). IBM Consulting can be engaged to assist in this effort; consult your Palo Alto Networks account team for assistance.
5) If the customer does not do step 4, having the log_source_mappings.out and qid_mappings.out files will allow the customer to extract/convert Ariel records without restoring the SRE-supplied configuration backup. The utilities will happily run on a vanilla install of QRadar. Again, this image can be run in a virtual environment with large resources allocated (vCPU intensive) to take advantage of massively parallel operations.

Components

The QRadar Ariel Export system is comprised of three main components, each with a distinct role in the data processing pipeline:

1. Extract PostgreSQL Info Perl Script (extract_psql_device_info.pl)

The `extract_psql_device_info.pl` script is a Perl script responsible for extracting essential metadata from the QRadar deployment's PostgreSQL database. This script generates two critical mapping files:

- **log_source_mappings.out**: Contains a comprehensive mapping of log source IDs to their names, device type IDs, device type names, and descriptions.
- **qid_mappings.out**: Contains mappings for QIDs (QRadar Identifiers) to their names and descriptions.

These files are crucial for enriching the Ariel record data during the conversion process, providing human-readable context to event IDs and log sources. The script can be run on any host in the QRadar deployment as the PostgreSQL database is replicated, and only needs to be run once, with resulting output files listed to be reused for different date ranges. For QRoC environments IBM SRE can run this script for you and deliver the resulting output files per the preparation steps above.

2. Ariel Dump Bash Script (ariel_dump.sh)

The `ariel_dump.sh` script orchestrates the entire export process. It is a Bash script designed to:

- **Export Ariel Records:** Utilizes QRadar's command-line Ariel export client (Java-based) to dump raw Ariel records.
- **Manage Concurrency:** Configured to handle concurrent Ariel Dump Client (Java) jobs and downstream Perl processes for data conversion using GNU Parallel.
- **Log Progress:** Maintains a log file to track the progress of the export.
- **Configuration:** Relies on external configuration files (`static_settings_ariel_dump.config` and `dynamic_settings_ariel_dump.config`) for user-tunable settings such as date ranges, base directories, and concurrency limits.

3. Process Ariel Export Perl Script (process_arielClientdump.pl)

The `process_arielClientdump.pl` script is a Perl script that takes Ariel dump data via standard input and transforms it into gzipped JSON files. Its primary functions include:

- **Ariel Record Parsing:** Reads raw Ariel records line by line and parses all 65 Ariel fields using a comprehensive regular expression.
- **Payload Decoding:** Decodes the `encoded_payload` field into a human-readable UTF-8 string.
- **Time Conversion:** Converts the `device_time_epoch_milliseconds` field into a ISO 8601 YYYY-MM-DD HH:MM:SS GMT timestamp.
- **JSON Record Generation:** Constructs a JSON hash containing selected parsed fields and integrates information from the `log_source_mappings.out` and optionally `qid_mappings.out` files, the latter of which allows organizations to maintain any custom QRadar Event ID's they may have developed.

- **File Output:** Organizes and outputs the processed JSON data into gzipped files, structured by year, month, and day (e.g., `<TLD>/YYYY/MM/DD/qradar-YYYY-MM-DD-<EventProcessorDesignator>-<LogSourceTypeName>.json.gz`). Files are compressed by default. The QRadar EventProcessorDesignator designation is contained within the file name for easy search selection criteria; customers will best know their environment.
- **Debugging:** Includes a debug mode for inspecting parsed fields of a single Ariel record.

Usage Instructions

The following diagram illustrates the high-level workflow of the QRadar Ariel Export process, starting with the extraction of PostgreSQL information:

This process outlines the steps for exporting QRadar Ariel data:

Step 1: PostgreSQL Data Extraction
A Perl script, "Extract QRadar PostgreSQL," gathers and consolidates information from the PostgreSQL database. This data is then saved into two files: `log_source_mappings.out` and `qid_mappings.out`. This component only needs to be run one time and can be reused during subsequent runs of the scripts for different date ranges.

Step 2: Ariel Database Dump
The "Ariel Dump Script," a Bash script, extracts QRadar Ariel DB records as text and outputs them to STDOUT. This script controls the entire process and provides for massive parallelization.

Step 3: Processing Ariel Client Dump
A Perl script, "Process Ariel Client Dump," takes the raw QRadar Ariel DB Records from Step 2 (as STDIN). It utilizes the output files generated in Step 1 as arguments for processing.

Step 4: Output Storage
The processed output from the "Process Ariel Client Dump" Perl script is stored as compressed JSON files. These files are organized on a daily basis, with one file per day for each combination of QRadar event-processor-designation / log-source-type.

Usage

1) Unpack the package of scripts to any location on the QRadar host.
2) Modify the static and dynamic configuration files as appropriate. It is recommended that you define no more than a week's worth of files to be

extracted/converted to test and achieve confidence.   At minimum the defined data range must be at least two days.

3) Ensure sufficient storage for resulting output files (compressed JSON).
4) Run the GNU Parallel script once to acknowledge their copyright notice (command: “bin/parallel –citation” or the logfile will show many invocations of GNU Parallel)
5) Within the utility ‘screen’ or ‘tmux’, execute the ‘ariel_dump.sh’ script. No output means it’s working without errors.   When all Ariel records are processed, the script will exit. Using ‘screen / tmux’ will ensure that the script continues to run even after a shell timeout event.   Upon shell timeout, simply log back into the system and restore the session with the appropriate ‘screen / tmux’ command option.
6) Keep an eye on three things, recommended to be running in three separate terminal sessions. A) Utility ‘top’ to monitor load average and process execution B) Watch the log file for progress (tail -f logfile_ariel_dump.log) and you will see how many Ariel records have been processed, completion percentage and how many Ariel records remain to be processed. C) Keep an eye on storage space where the compressed JSON output files are being written. This is key to finishing the processing run.   If you run out of storage space, you may have to delete all the output files and start over after you solve the space issue.   Files are not processed sequentially in date/time order as there is an effort to pull Ariel records from any random time during the entire processing date range in order to distribute the I/O load among spindles.
7) As the programs are running, adjust the settings in the `‘dynamic_settings_ariel_dump.config`’ file to increase concurrency. Through benchmark testing it has been derived that 4 Concurrent Data Conversion Jobs per Ariel Dump Job is a good ratio to maintain and should not be adjusted without due deliberation   However, the number of Ariel Dump Jobs is directly related to the amount of (v)CPU’s available in the QRadar host.   Continue to gradually increase the amount of Ariel Dump Jobs until the utility ‘top’ shows either A) a load average of (2 * number of (v)CPU’s) or B) in utility ‘top’ toggle to per-cpu view by hitting the number ‘1’ and then watch until the per-cpu utilization goes to low-to-mid 90% utilization. In either case, there should be no I/O wait time incurred.
8) Rinse, lather and repeat with modified date ranges until completion of extraction/conversion.

If you encounter code issues, please email me with a description of problems encountered and some screen example snippets and I will provide you with best effort support for coding issues.

For consulting help with extraction of QRadar Ariel Data, reach out to your Palo Alto Networks account team.