# *Velociraptor*


CSE 406: Computer Security Sessional

Faria Binta Awal: 1905012
Fatema Tuj Johora: 1905022

Bangladesh University of Engineering and Technology

March 9, 2024

# Contents

# Chapter 1

# Introduction to Velociraptor

## 1.1 What is Velociraptor


Velociraptor is an open-source digital forensics and incident response (DFIR) tool designed for comprehensive endpoint visibility and analysis. It serves as a powerful framework for collecting and analyzing data from endpoints to facilitate threat hunting, incident detection, and forensic investigations. Below is an overview of Velociraptor's key features and functionalities:

**Key-Benefits:**

- **Endpoint Visibility:**
  Velociraptor is focused on providing deep visibility into endpoints within a network. It allows users to collect a wide range of information from individual devices, including file metadata, process details, registry entries, and more.
- **Rule-Based Detection and Threat Hunting:**
  Rule-Based Detection and Threat Hunting: Leveraging a rule-based detection engine, Velociraptor enables the creation and customization of detection rules. Security professionals can define rules to identify specific indicators of compromise (IoCs) or patterns associated with malicious activities, allowing for effective threat hunting.

- **Artifact Collection:**

  Velociraptor supports the collection of artifacts from endpoints. Artifacts may include file system data, registry information, event logs, and other digital artifacts crucial for forensic analysis. The tool's modular architecture allows the easy addition of custom artifact collectors.

- **Scalable and Agentless:**
  Velociraptor is designed to be scalable and operates in an agentless manner. The tool's server-client architecture enables centralized management, making it suitable for large-scale enterprise environments.

- **Forensic Analysis:**
  Velociraptor facilitates forensic investigations by providing detailed information about endpoint activities. It allows investigators to reconstruct events, identify the timeline of activities, and gather evidence necessary for understanding and responding to security incidents.

- **Timeline Analysis:**
  The tool supports timeline analysis, allowing security professionals to visualize and correlate events over time. This feature aids in understanding the sequence of activities on an endpoint and identifying potential malicious behavior.

- **Customizable Dashboards and Reports:**
  Velociraptor offers customizable dashboards and reporting capabilities. Security teams can generate reports on endpoint activities, analyze trends, and present findings in a format suitable for various stakeholders.

- **Community-Driven Development:**
  Being an open-source tool, Velociraptor benefits from community-driven development and collaboration. Users can contribute to the tool's development, share detection rules, and participate in the improvement of its capabilities.

In summary, Velociraptor serves as a robust solution for endpoint visibility, detection, and forensic analysis. Its modular and customizable nature makes it adaptable to diverse environments, and its open-source model encourages continuous innovation and collaboration within the cybersecurity community.

## 1.2 Installation

The installation of Velociraptor varies depending on the operating system in use. However, for demonstration and educational purposes, we focus on using Windows

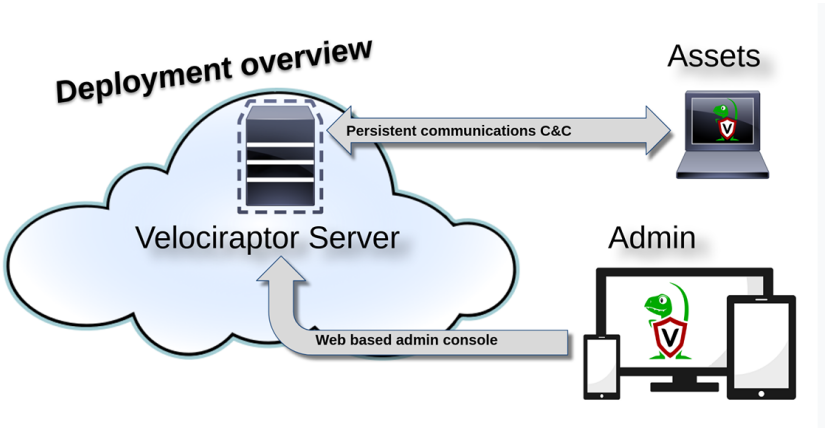
**Steps For Installation**

- At first, download the velociraptor from it's release page of github .
- We downloaded the version with .exe format
- Then the tool is set up

## 1.3 Deployment

### 1.3.1 Overview

Below is a typical Velociraptor deployment




**Major parts include:**

- **Client:** The client is the instance of the Velociraptor agent running on the endpoint.
- **Frontend:** The frontend is the server component communicating with the client.
- **Gui:** The gui is the web application server that presents the administrative interface.
- **Server:** The API server is used to accept API requests.

### 1.3.2 Instant Velociraptor

We can deploy server and client in a different way in our local machine. If a Velociraptor sandbox is desired for evaluation, testing, or other purposes, Instant Velociraptor can be installed. A fully functional Velociraptor system is deployed solely to the local machine. The Velociraptor executable for the respective platform can be downloaded from the GitHub project's releases page, and the "gui" command can be executed to run the system.

```
C:\Program Files\Velociraptor>Velociraptor.exe gui
```

This will create a new server configuration and start a new server on the local machine. It will also start a local client communicating with the server.

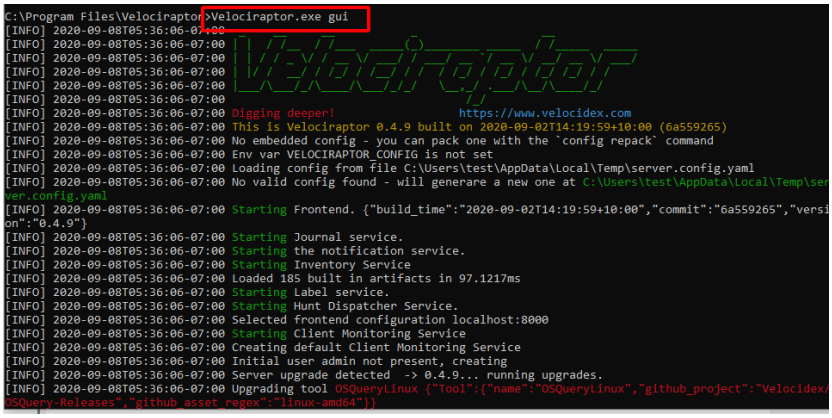



Figure 1.1: Initiate GUI

- The Server only listens on the local loopback interface.
- The Client connects to the server over the loopback. A data store directory is set to the user's temp folder. A single administrator user is created with username admin and password. A browser is launched with those credentials to connect to the welcome screen.

### 1.3.3 Self-Signed SSL

Velociraptor deployments are secured using a self-signed Certificate Authority (CA) that is generated during the initial configuration generation step. The client's configuration contains the signed CA, which is used to verify all certificates needed during communications.

In self-signed SSL mode, Velociraptor issues its own server certificate using its internal CA. This means the Admin GUI and front end also use a self-signed server certificate.

**Server Connection**

If a Velociraptor sandbox is desired for evaluation, testing, or other purposes, Instant Velociraptor can be installed. A fully functional Velociraptor system is deployed solely to the local machine. The Velociraptor executable for the respective platform can be downloaded from the GitHub project's releases page, and these commands can be executed to run the system. To run the server, at first we have to Navigate to the Velociraptor Directory.

- **Generate The configuration File:**

```
C:\Program Files\Velociraptor>Velociraptor.exe config generate -i
```

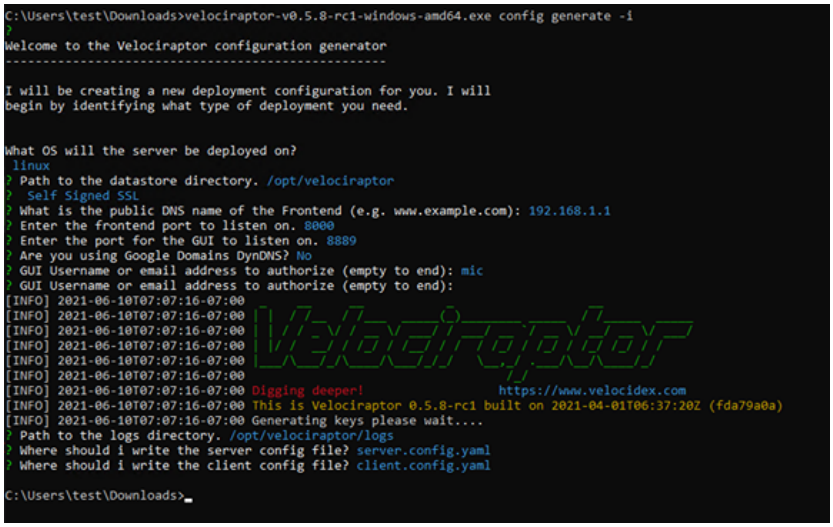
The configuration wizard appears.
The configuration wizard includes a set of questions to guide you through the first step of the deployment process.

- **Create the server package:**

```
C:\Program Files\Velociraptor>Velociraptor.exe --config
server.config.yaml user add admin --role administrator
```

- **Deploy Server:**

```
C:\Program Files\Velociraptor>Velociraptor.exe --config
server.config.yaml frontend -v
```

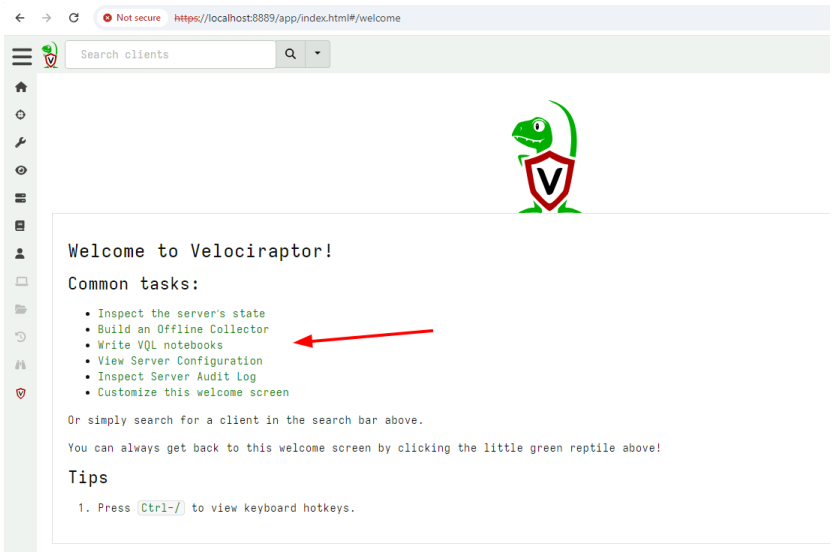



**Deploy Clients**

To run the client, at first we have to Navigate to the Velociraptor Directory and run the server. [**Note:** Velociraptor is run on our local machine. The local setup allows only one client to be connected at a time, and our client number is 1.]

```
C:\Program Files\Velociraptor>Velociraptor.exe --config
client.config.yaml client -v
```

**After deployment , you will see the Admin Gui**




The Admin GUI is a web application that can be used to interact and manage Velociraptor. The GUI allows users to schedule new collections, edit existing artifacts or write new ones and launch hunts.

# Chapter 2

# Overview of Source Code

## 2.1 Architecture Overview

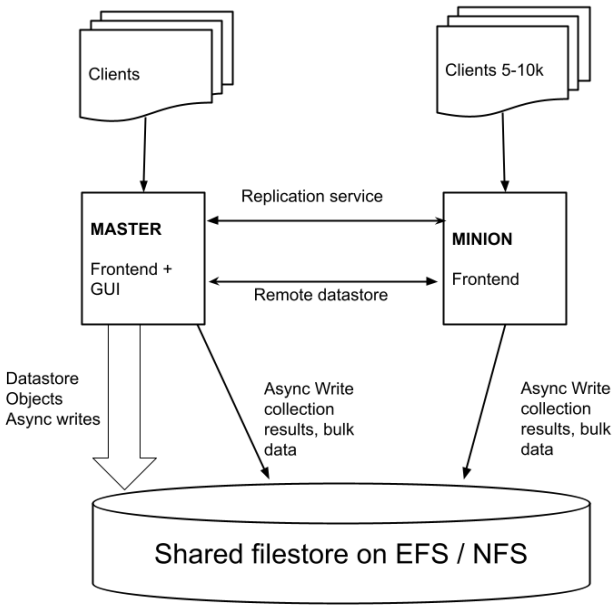



Figure 2.1: Multi Frontend architecture overview

## 2.2 Source Code Overview

The source code of Velociraptor is structured into multiple directories, each serving a different aspect of its functionality:

- **.github/workflows**: Contains GitHub action workflow files for automated testing and building processes.
- **accessors**: Related to the data access layer.
- **acls**: Deals with Access Control Lists.
- **actions**: Contains code for actions like data collection.
- **api**: Contains the application programming interface definitions.

- **artifacts**: Holds the VQL definitions for data collection artifacts.
- **bin**: Typically contains compiled binaries or scripts.
- **config**: Contains configurations for Velociraptor's operation.
- **constants**: Houses constant values used throughout the codebase.
- **crypto**: Contains cryptographic functionalities.
- **datastore**: Manages data storage.
- **debian**: Related to Debian-based Linux distributions.
- **docs**: Documentation files.
- **executor**: Related to the execution of VQL queries.
- **file_store**: Manages file storage.
- **flows**: Logic related to data collection and analysis flows.
- **glob**: File globbing functionalities.
- **grpc_client**: Related to the gRPC client implementation.
- **gui**: Contains the graphical user interface components.
- **http_comms**: Handles HTTP communications.
- **json**: Involves JSON processing utilities.
- **logging**: Manages logging functionalities.
- **notifications**: Handles notification management.
- **paths**: Deals with filesystem paths.
- **proto**: Contains Protocol Buffer files.
- **reporting**: Involves the generation and management of reports.
- **responder**: Related to response handling.
- **result_sets**: Manages sets of results produced.
- **scripts**: Contains various scripts.
- **server**: Manages server-side functionalities.
- **services**: Contains service definitions and implementations.
- **startup**: Code related to initialization and startup.
- **third_party**: Holds third-party libraries or code.
- **timelines**: Related to time-based data or functionalities.
- **tools**: Houses additional tools or utilities.
- **uploads**: Manages file uploads.
- **utils**: Contains utility functions and helpers.
- **vql**: Related to Velociraptor Query Language files.
- **vql_plugins**: Plugins for extending VQL capabilities.

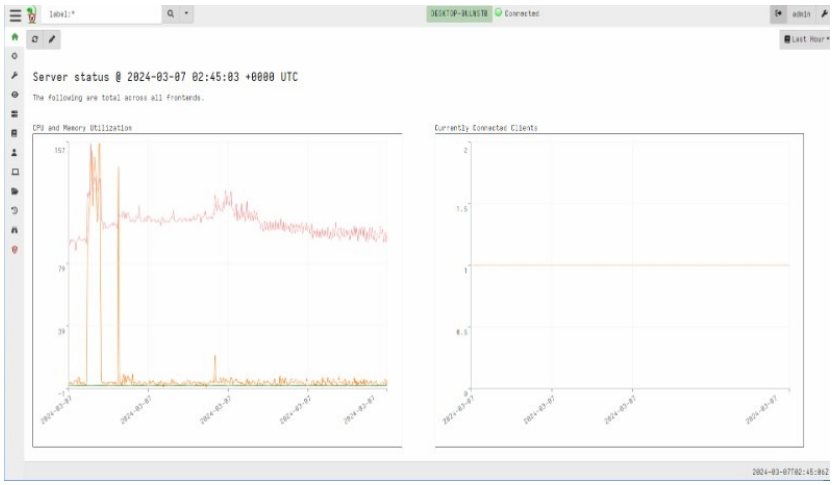
# Chapter 3

# Tool Overview

## 3.1 The Admin Gui

### 3.1.1 Homepage

**DashBoard:** The Dashboard can be accessed from the Home icon on the sidebar. The dashboard shows the current state of the deployment at a high level.



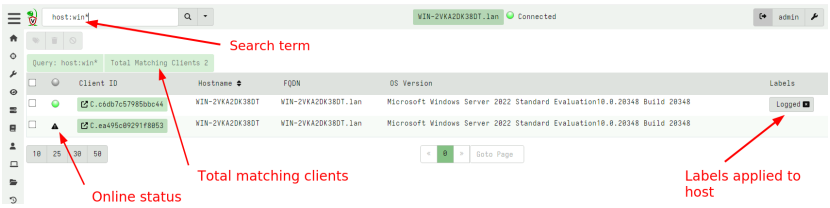

The dashboard is divided into two parts. On the left, the total memory and CPU used by all frontends is down over the past day. On the right, the total number of currently connected clients is shown.

### 3.1.2 Inspect Clients

Velociraptor clients are endpoints with the Velociraptor agent running on them. Since Velociraptor maintains a persistent connection to the server, each endpoint is immediately available to interact with.
Typically we begin our investigation by searching for a client, selecting it, and interactively collecting artifacts from it.

### 3.1.3 Search Clients

Below is a typical Velociraptor deployment




**The table contains five columns:**

- **State:**The online state of the host is shown as a color icon. A green dot indicated that the host is currently connected to the server, a an exclamation icon indicates the host is not currently connected but was connected between 15 minutes and 24 hours ago. A warning triangle indicates that the host has not been seen for 24 hours or more
- **ClientId:** The client ID of the host is shown. Clients have a unique ID starting with C.. Internally, the client ID is considered the most accurate source of endpoint identity. Velociraptor always refers to the client ID rather than the hostname because hostnames can change. A client ID is derived from the client's cryptographic key and is stored on the endpoint in the client writeback file. Clicking on the client id will take you to the host information screen.
- **Hostname:** The hostname and Fully Qualified Domain Name reported by the host.
- **OS version:** The operating system version. This indicates if the host is a Windows/Linux/MacOS machine and its respective version.
- **Label:**Any labels applied to the host. Clicking on a label removes the label from this host.

### 3.1.4 Select a Client

Click on any client in the search screen to view information relevant to the selected client.

You can easily tell which client we are dealing with, as the name of the host and the last time we connected with it are shown:

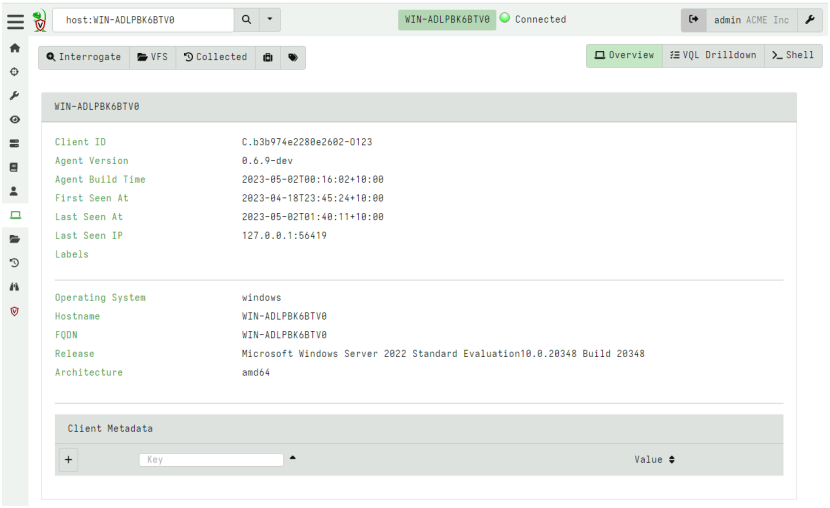



Figure 3.1: Client Overview

Velociraptor maintains some basic information about the host, such as its hostname, labels, last seen IP, and last seen time. This is shown in the Host View pane. Velociraptor uses this information to make it possible to search for this host and target it for further investigation. Velociraptor gathers this information constantly from the endpoint and upon first enrollment, so this information should be relatively up to date. You can refresh this information at any time by clicking the Interrogate button.

## 3.2 VQL

### 3.2.1 What is VQL

VQL is central to the design and functionality of Velociraptor, and a solid grasp of VQL is critical to understanding and extending Velociraptor.

### 3.2.2 Why a new Query Language?

The need for a query language arose from our experience of previous Digital Forensic and Incident Response (DFIR) frameworks. Endpoint analysis tools must be flexible enough to adapt to new indicators of compromise (IOCs) and protect against new threats. While it is always possible to develop new capability in code, it's not always easy or quick to deploy a new version.

A query language can accelerate the time it takes to discover an IOC, design a rule to detect it, and then deploy the detection at scale across a large number of hosts. Using VQL, a DFIR investigator can learn of a new type of indicator, write relevant VQL queries, package them in an artifact, and hunt for the artifact across the entire deployment in a matter of minutes.

Additionally, VQL artifacts can be shared with the community and facilitate a DFIR-specific knowledge exchange of indicators and detection techniques.

### 3.2.3 VQL Structure

Let's consider the basic syntax of a VQL query.


Figure 3.2: VQL Structure

The query starts with a SELECT keyword, followed by a list of Column Selectors then the FROM keyword and a VQL Plugin potentially taking arguments. Finally we have a WHERE keyword followed by a filter expression.

**Plugin**

While VQL syntax is similar to SQL, SQL was designed to work on static tables in a database. In VQL, the data sources are not actually static tables on disk - they are provided by code that runs to generate rows. VQL Plugins produce rows and are positioned after the FROM clause.

Like all code, VQL plugins use parameters to customize and control their operations. VQL Syntax requires all arguments to be provided by name (these are called keyword arguments). Depending on the specific plugins, some arguments are required while some are optional

### 3.2.4 Life of a Query

In order to understand how VQL works, let's follow a single row through the query.

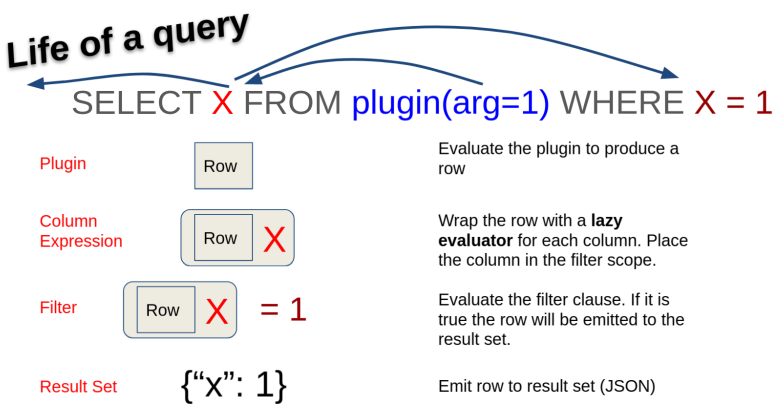



Figure 3.3: Life of a query

## 3.3 NoteBook

### 3.3.1 What is Notebook

Notebooks are interactive collaborative documents which can interleave markdown and VQL queries in to create an interactive report. Notebooks are typically used to track and post process one or more hunts or collaborate on an investigation.

### 3.3.2 Create a Notebook

**Steps:**

- Start the Velociraptor GUI. This can be easily done by ” velociraptor.exe gui” command
- Select Notebooks from the sidebar menu then “Add Notebook” .
- Give the notebook a name and a description and submit. The new notebook is created

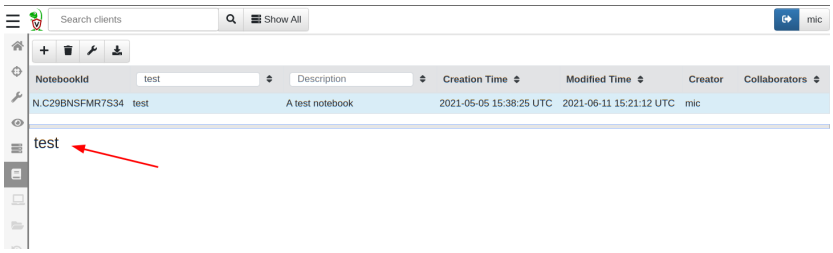



Figure 3.4: Notebook Creation

### 3.3.3 Edit Notebook

**Steps:**

- Click on the cell to give it focus and the cell control toolbar will be shown, from here click the Edit Cell button to edit the cell contents.

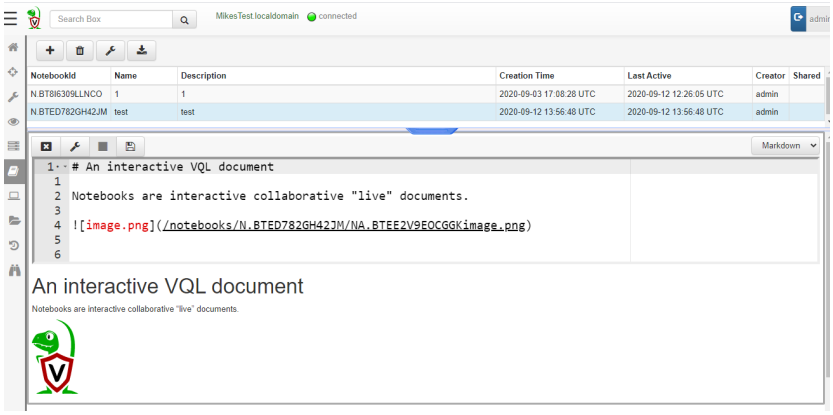



Figure 3.5: Edit Notebook

- Lets add a new cell to the notebook. Click the Add Cell button and a pull down menu appears offering the type of Cell that can be added. For now, select a VQL cell.

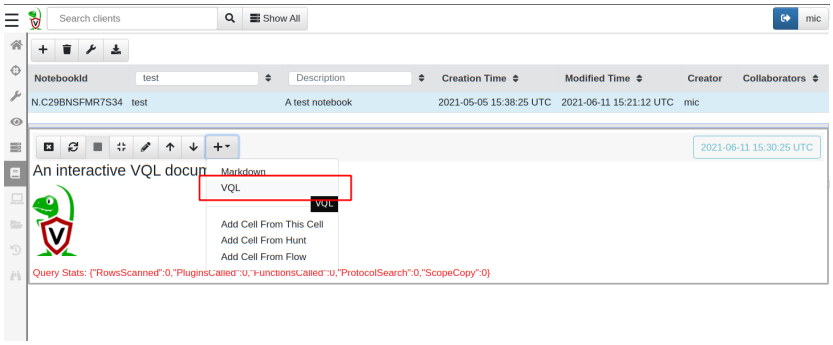



Figure 3.6: Add a new Cell

- After clicking the Edit Cell button, you can type VQL directory into the cell. As you type, the GUI offers context sensitive suggestions about what possible completions can appear at the cursor. Typing "?" will show all suggestions possible.

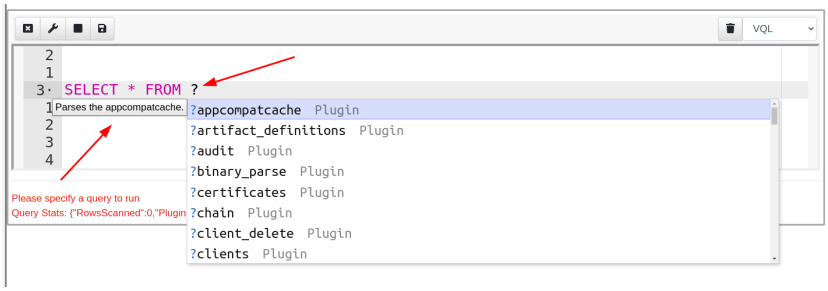



Figure 3.7: Add a new Cell-2

### 3.3.4 Share Notebooks

By default, notebooks are set to private for the user who created them. When a notebook is created or edited, the option to share it with all users can be activated by clicking the Public checkbox. Additionally, it can be shared selectively with specific users by choosing their names in the Collaborators field.

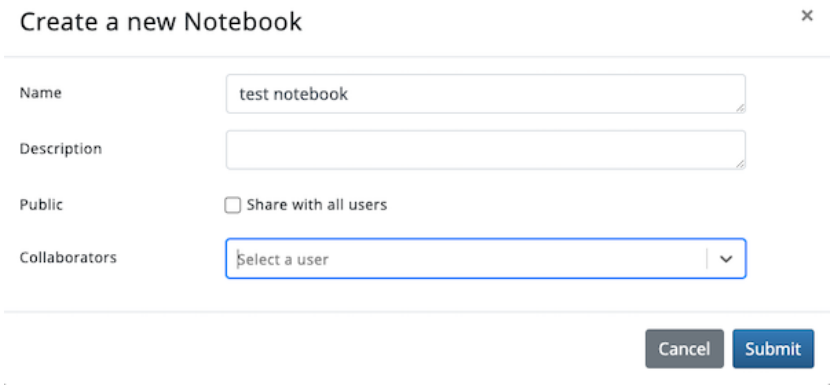



Figure 3.8: Share Notebooks

## 3.4 Artifacts

### 3.4.1 What is Artifact

An Artifact is a way to package one or more VQL queries in a human readable YAML file, name it, and allow users to collect it. An Artifact file simply embodies the query required to collect or answer a specific question about the endpoint.
Ultimately Velociraptor is simply a VQL engine . That is, it processes a VQL query producing a series of rows and sends those rows to the server.

### 3.4.2 Why Artifact

- **Interactively investigating an endpoint:**Although the VFS presents a familiar interface, it is not ideal for quickly finding the files and registry keys we are usually interested in. One would need to know exactly which files are of interest and then click over multiple directories searching for these files. To automate collection it is better to write special purpose VQL Artifacts to identify the information of interest.

- **Easy to use**:Artifacts encapsulate a VQL query inside a YAML file so that end users do not need to understand the query to use it. This makes artifacts easier to use and facilitates knowledge sharing with more experienced users.

### 3.4.3 View Artifacts

**Steps:**

- Select "View Artifacts" from the left side bar

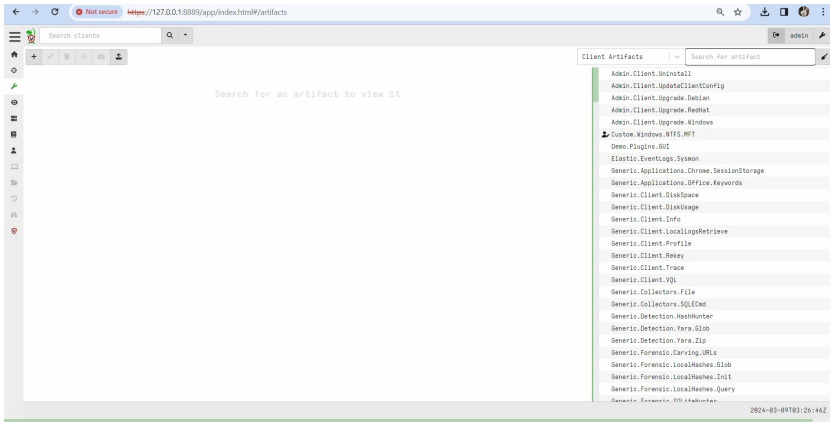



Figure 3.9: View Artifact

- Then Select the required artifact to know details about it

### 3.4.4 Create Customized Artifacts

**Steps:**

- Select "+" icon ,to add a new customized artifact

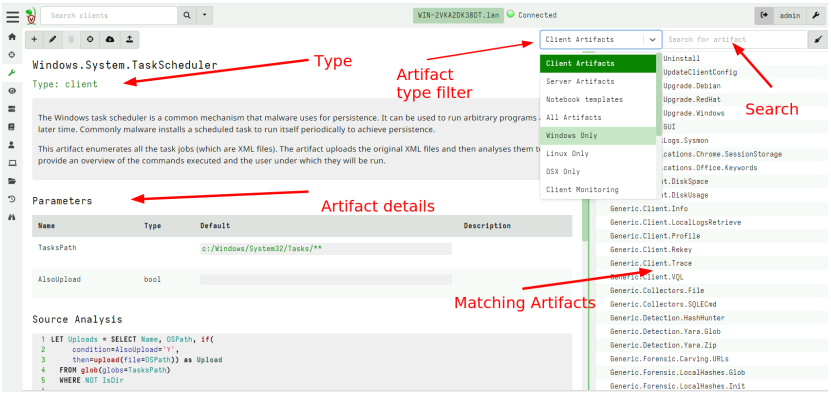



Figure 3.10: View Details of an Artifact


Figure 3.11: Add a new Artifact

- Then click save to save the artifact

### 3.4.5 Edit Artifact

Artifacts can be edited by clicking the “Edit an Artifact” button

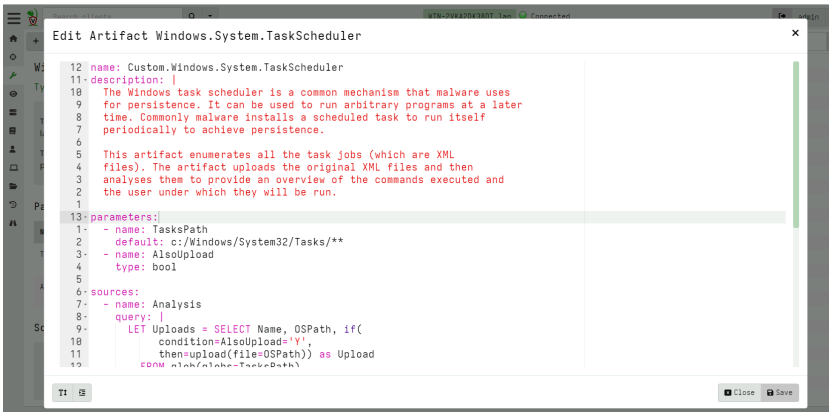



Figure 3.12: Edit Artifacts

Then save it

### 3.4.6 Collect Artifacts

**Steps:**

- Start a new collection by clicking the New Collection button . This will open the new collection wizard as show below.

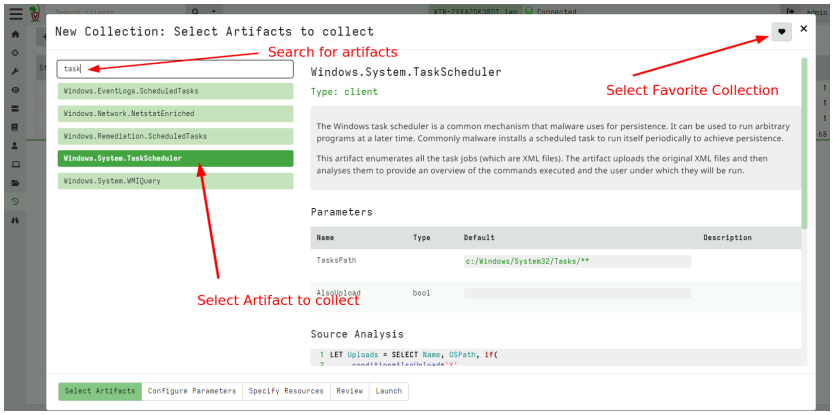



Figure 3.13: Create a new Collection

- Then click save to save the artifact

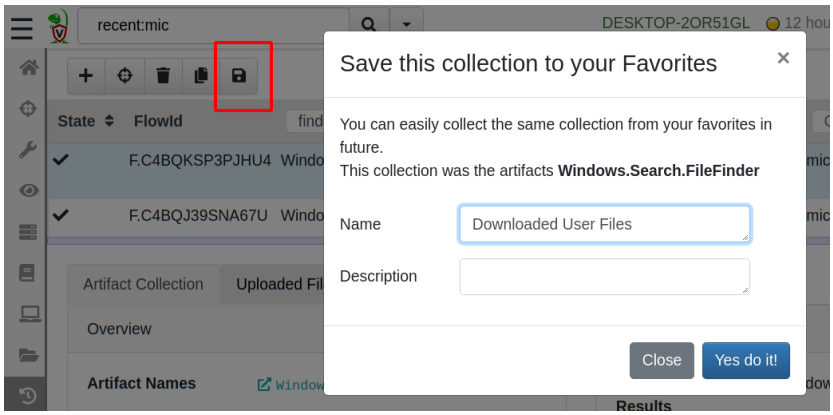



Figure 3.14: Add to Favorites

- The next step allows us to modify artifact parameters.

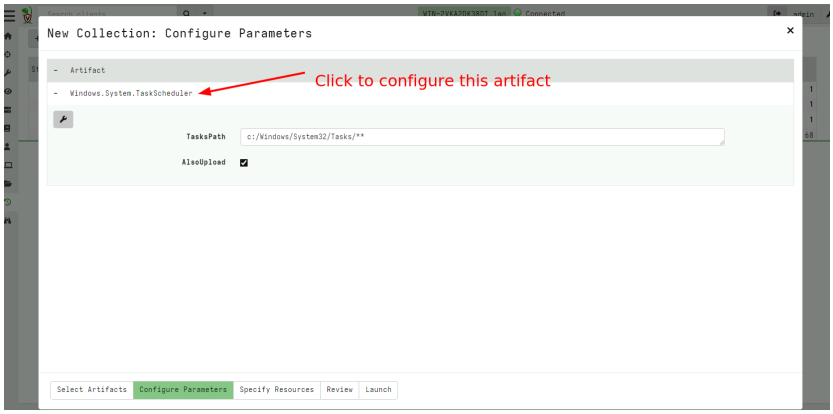



Figure 3.15: Modify Parameters

- Each Artifact parameter has a default value. For the purposes of our example, we will upload some raw XML files. Click Launch to start the collection. After a short time, the collection will complete.

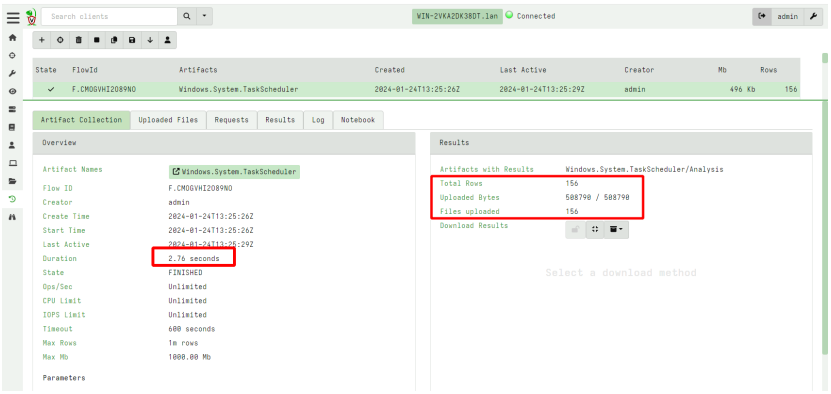



Figure 3.16: View Collection

We can see that this collection uploaded 195 files and added 195 rows. The VQL query parses each XML file in turn and uploads it.
We can see more information about this collection in the tabs in the bottom pane.

**Collection Details:**

- **Logs -** As the VQL query is executing on the endpoint, the query may produce log messages. This is called the Query Log and it is forwarded to the server. We are able to see how the query is progressing based on the query log.

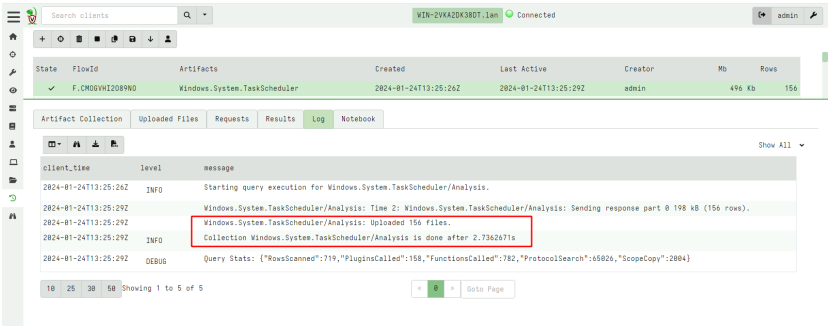



Figure 3.17: Query Logs

- **Uploaded Files -** This tab shows all the files uploaded by this query. You can download any of these files individually from the server by simply clicking the link, or click the preview button to examine the file in the GUI.

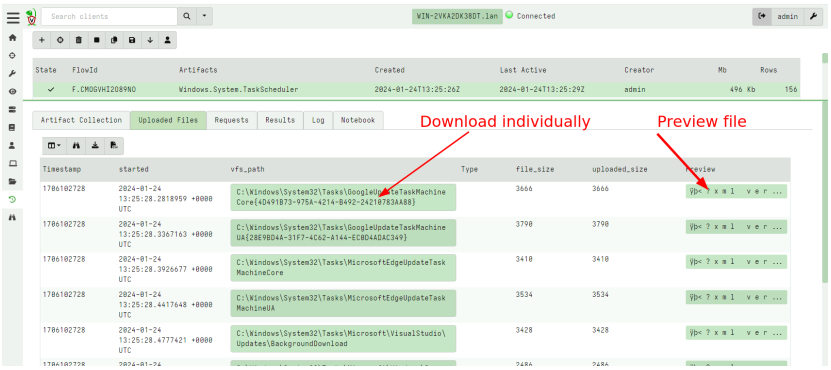



Figure 3.18: Uploaded Files

- **Result Tab -** This shows each result set in a table. A single collection may collect several artifacts. In this case you can choose which artifact to view by clicking the pull down menu.

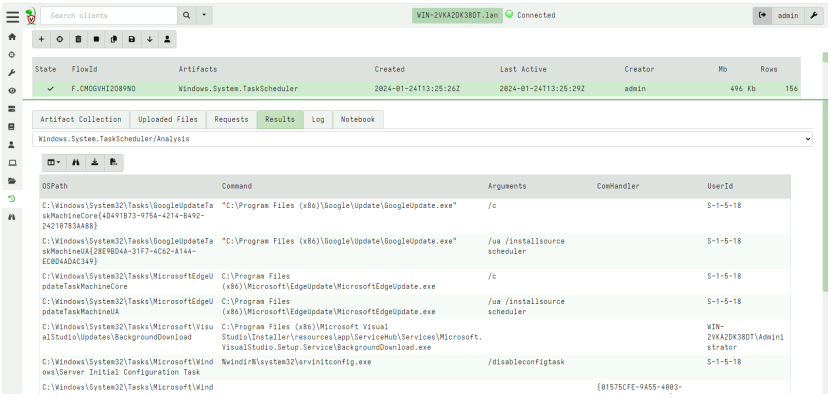


Figure 3.19: Result Tab

## 3.5 Hunting

### 3.5.1 What is Hunt

A hunt is a logical collection of a one or more artifacts from a group of systems. The Hunt Manager is a Velociraptor component that is responsible for scheduling collections of clients that met certain criteria, then keep track of these collections inside the hunt.
The important takeaway from this is that artifacts are still collected from endpoints the same way as we did previously, it is simply automated using the hunt manager.

### 3.5.2 Why Hunt

With Velociraptor, you can collect the same artifact from multiple endpoints at the same time using a Hunt. Hunts allow you to do the following:

- Monitor offline endpoints by scheduling hunts collecting artifacts from any endpoints that come back online during a certain period.
- Keep track of which endpoints collected the artifact and make sure the same artifact is not collected more than once on any endpoint.
- Examine the results from all collections easily.

### 3.5.3 Schedule a new hunt

**Steps:**

- select the "Hunt Manager" from the sidebar
- Then select "New Hunt" button to see the New Hunt Wizard.

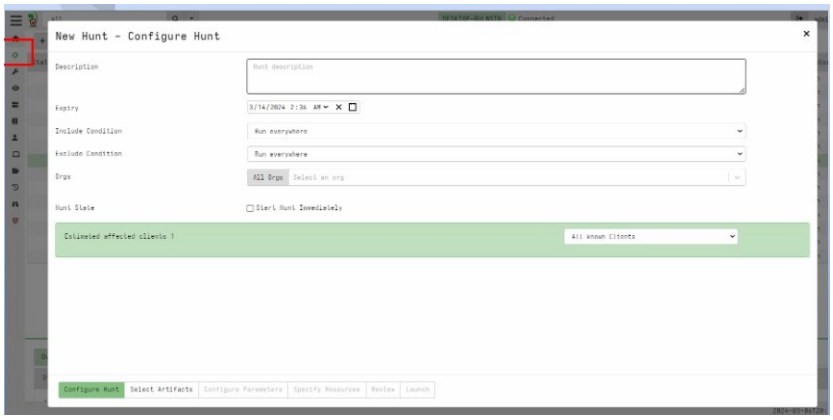



Figure 3.20: Create Hunt

Provide the hunt with a description and set the expiration date. You can also target machines containing the same label (A Label Group), or exclude the hunt from these machines.

- Next you need to select and configure the artifacts as before. Once everything is set, click Launch Hunt to create a new hunt.

### 3.5.4 Run a Hunt

**Steps:**

- Hunts are always created in the Paused state so you will need to click the Start button before they commence. Once a hunt is started many hundreds of machines will begin collecting that artifact, be sure to test artifact on one or two endpoints first.

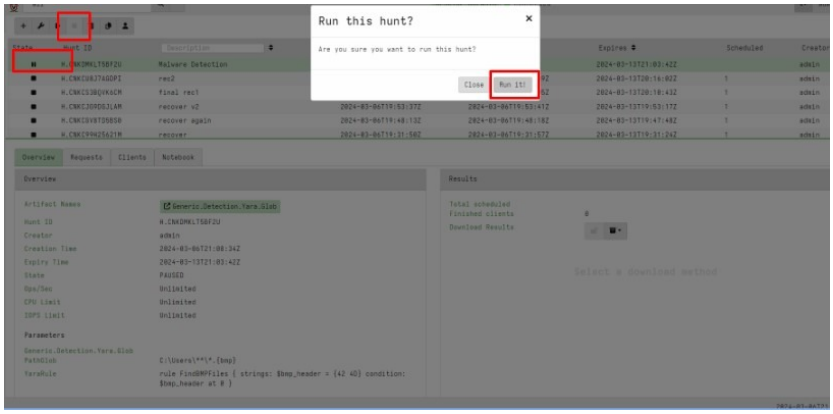



Figure 3.21: Run the Hunt

- The hunt's progress can be monitored. As clients are scheduled they will begin their collection. After a while the results are sent back and the clients complete.

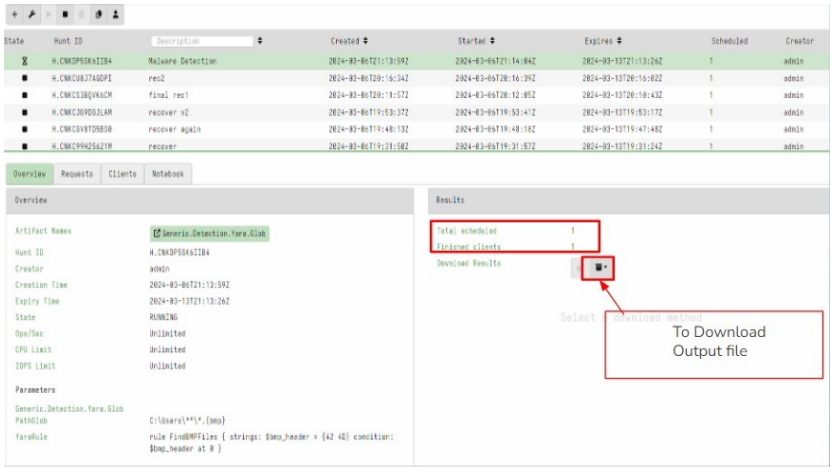



Figure 3.22: View Progress of the Hunt

## 3.6 VFS

### 3.6.1 What is VFS

In Velociraptor, "VFS" stands for "Virtual File System." The Velociraptor Virtual File System is a core component of the Velociraptor framework, and it plays a crucial role in collecting and analyzing data from endpoints.

### 3.6.2 Why VFS

- VFS provides a consistent interface for accessing endpoint data.
- Ensures seamless operation across diverse operating systems.
- Supports thorough analysis of file attributes and system activities.
- Gathers valuable artifacts aiding in identifying indicators of compromise.
- Enables reliable data collection in environments of varying scales.

### 3.6.3 View VFS

The VFS consists of a tree view in the left pane and a file listing in the top right pane. The tree view allows us to navigate through the filesystem, starting at the top level. Remember that the GUI is simply viewing data that was previously collected from the client. When clicking on a directory in the tree view that has not been synced from the client yet, the top right pane shows the message No data available. Refresh directory from client by clicking above..

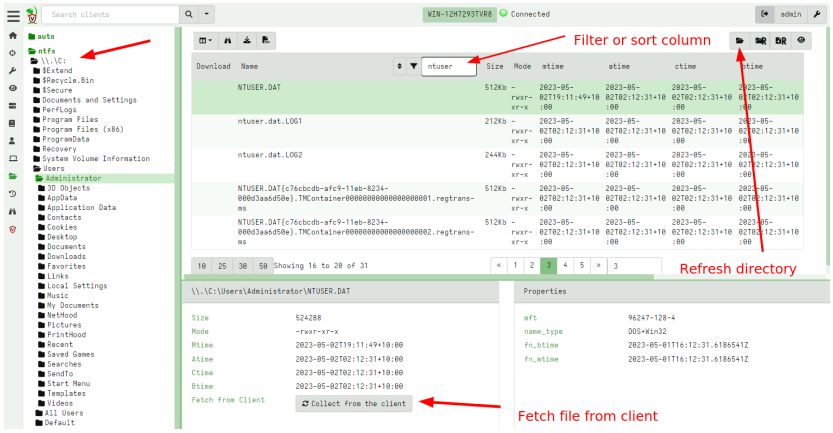



Figure 3.23: View VFS

Clicking on the refresh directory button will initiate a directory listing operation on the client, and providing the client is currently connected, will refresh the VFS view. Similarly the recursive refresh directory button will recursively refresh the directory listing from the current directory down.

### 3.6.4 Preview a file

Once a file is fetched from the endpoint it is stored on the server. You can quickly preview the file in the GUI by clicking on the preview button (initially the preview button shows the first few characters from the file, which helps to quickly eyeball the file type).

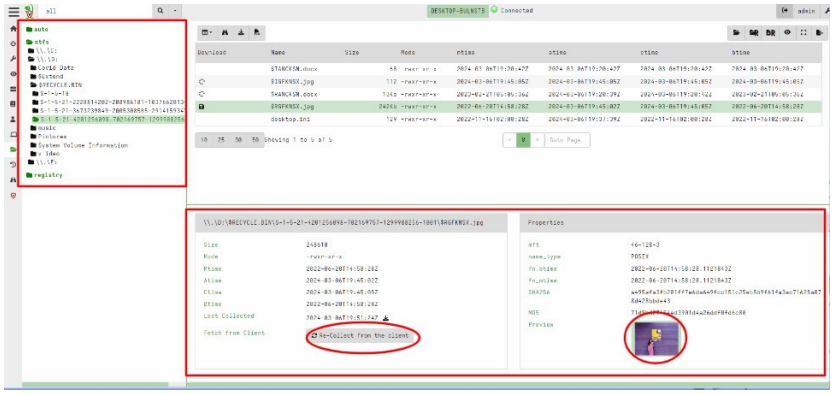



Figure 3.24: Preview Files-1

The Preview screen is a hex viewer with some useful features:

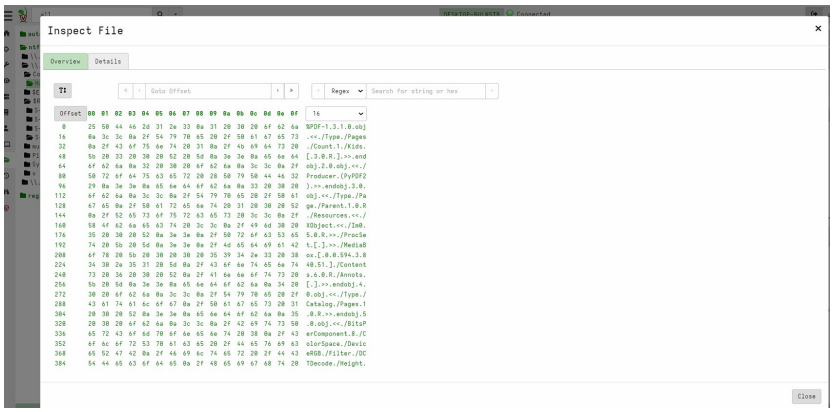



Figure 3.25: Preview Files-2

- **Text View:** View a text only version of the data (this removes non printable characters from the binary data and shows only ASCII strings).
- **Goto Offset:** allows to skip to arbitrary offsets in the file
- **Search file:** allows to search the file using Regex, String or Hex String modes.

# Chapter 4

# Features

## 4.1 Overview

Velociraptor includes the following features:

- Scalable data collection
- Advanced query capabilities with VQL (Velociraptor Query Language)
- Artifacts repository for predefined data collection scripts
- Real-time monitoring and alerting for incident response
- Built-in GUI for easy management and analysis
- Support for various platforms including Windows, Linux, and macOS
- Encrypted communications for secure data transfer
- Extensible architecture allowing for custom artifacts and integrations
- Malware detection
- Recover deleted files
- Web browser history analysis
- Continuous monitoring and detect unsuccessful logins

Here, we have discussed four features.

## 4.2 Short Description

Velociraptor is a versatile forensic and monitoring tool designed for digital investigations and real-time analysis. Discussed four features are:

- **Malware Detection**: Velociraptor utilizes advanced query capabilities through its Velociraptor Query Language (VQL) to detect indicators of compromise and malicious activities. By leveraging its rich set of predefined artifacts, users can search for known malware signatures, anomalous process behaviors, and other signs of compromise across all endpoints in real-time.
- **Recover Deleted Files**: Leveraging the forensic capabilities of Velociraptor, users can recover deleted files from endpoints. This feature is particularly important in forensic investigations, allowing investigators to retrieve lost or intentionally deleted data, which can be crucial evidence.

- **Web Browser History Analysis**: Velociraptor can analyze web browser history across different platforms and browsers. This enables investigators to review visited websites, search history, and other internet-related activities, which can provide valuable context and evidence in digital investigations.
- **Continuous Monitoring and Detect Unsuccessful Logins**: Velociraptor provides continuous monitoring of endpoints, enabling real-time detection of suspicious activities, including repeated unsuccessful login attempts. This feature helps in identifying potential brute force attacks or unauthorized access attempts, contributing to the proactive defense mechanisms of an organization.

# Chapter 5

# Documentation of Features

## 5.1 Malware Detection

In velociraptor, Malware can be detected in many ways. Here, we used **Yara** to detect malwares.

### 5.1.1 Yara

YARA is an open-source tool designed for identifying and classifying malware samples based on textual or binary patterns. It was developed by Victor Alvarez of VirusTotal and is widely used in the cybersecurity community. The name "YARA" stands for "Yet Another Recursive Acronym," highlighting the tradition in the cybersecurity field of creating acronyms that refer to themselves
**Yara Rules:**

- YARA rules are the heart of YARA's functionality. They are written in a custom syntax and define patterns that can be used to identify specific characteristics or behaviors within files.
- Rules consist of a series of conditions and can include strings, regular expressions, and other elements that describe the characteristics of malware.
- YARA rules are written in a human-readable format and can be customized based on the specific needs of the analyst or security professional.

**Why use Yara**

- YARA identifies malware by matching specific patterns in files.
- Create rules tailored to detect unique malware characteristics.
- Incorporate threat intelligence for targeted and informed malware detection.
- YARA enhances analysis efficiency by pinpointing malware indicators accurately.
- Quickly adapt YARA rules to detect emerging and evolving malware.

### 5.1.2 Used Artifact

Used artifact for this feature is **Generic.Detection.Yara.Glob**.
"Generic.Detection.Yara.Glob" is a broad identification of potential threats using YARA rules and glob (wildcard) patterns. This method, employed by security software, detects suspicious patterns indicative of malware, providing a generic alert for further investigation.

### 5.1.3 Detect Malware using Yara

**Steps:**

- select the "Hunt Manager" from the sidebar

- Then select “New Hunt” button to see the New Hunt Wizard.

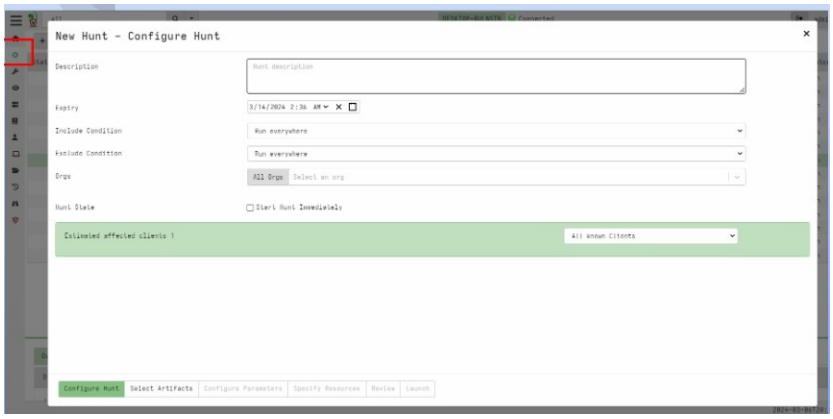



Figure 5.1: Create Hunt

- Next you need to select and configure the artifacts as before. Select “Generic.Detection.Yara.Glob” artifact
- Modify the parameters

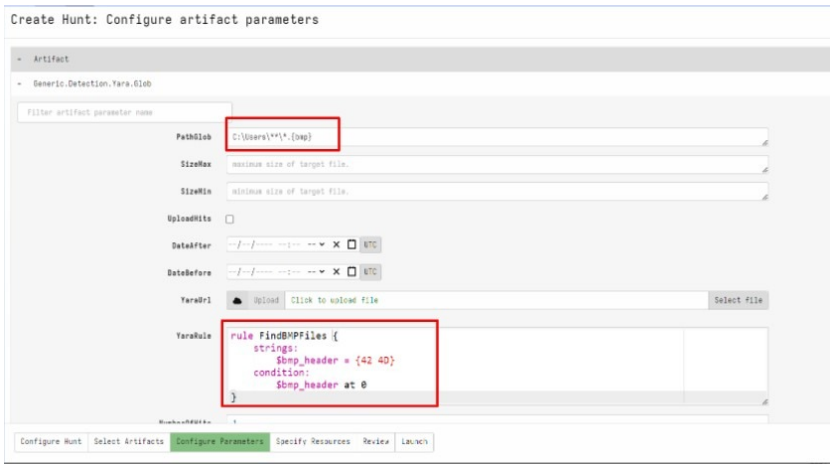



Figure 5.2: Modify Parameters

- Launch the hun and run it

### 5.1.4 Output and Results

**Steps:**

- After a while the results are sent back and the clients complete.

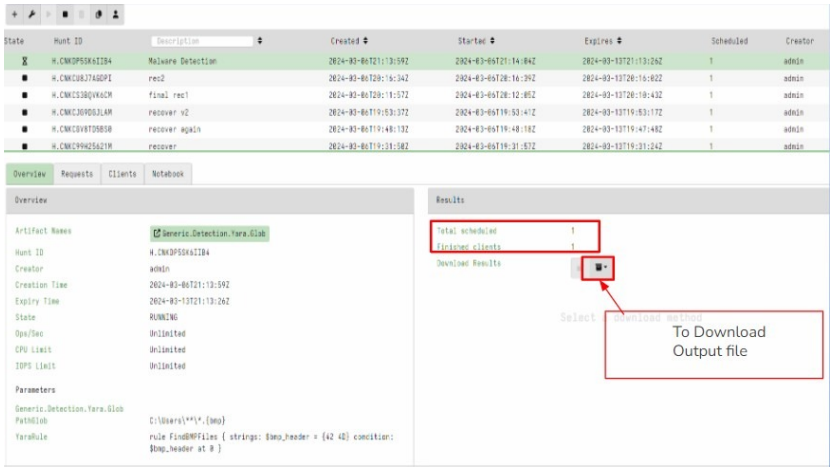



Figure 5.3: View Progress of the Hunt

- Now check the output in notebook section.

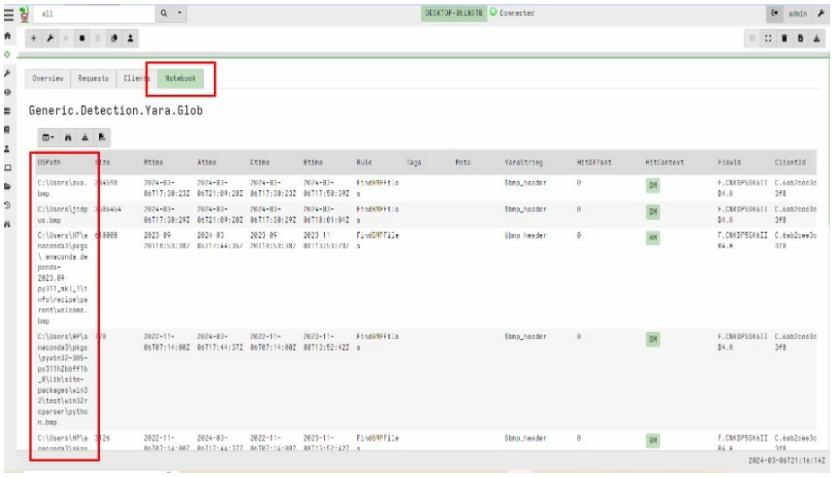



Figure 5.4: See Result

Here, the detection of all BMP files is observed as YARA rules have been configured to identify this file type. In this context, BMP files are being considered as potential indicators of malicious activity. Similarly, by setting up rules to match patterns specific to known malware, YARA provides the capability to identify and classify files based on their characteristics, aiding in the detection of potential security threats.

## 5.2 Recover Deleted Files

In velociraptor, deleted files can be traced and recover also using inode.

### 5.2.1 Inode

An inode, short for "index node," is a data structure in Unix-like file systems that uniquely identifies and stores metadata about files and directories. It contains essential information such as file type, permissions, owner details, timestamps, and pointers to the data blocks where the file's content is stored. Inodes enable efficient file access by separating metadata from file names and content. They support features like hard links, where multiple filenames can point to the same inode, and play a critical role in filesystem consistency, recovery, and disk space management. Understanding inodes is crucial for effective file system administration on Unix-like operating systems.

### 5.2.2 Recover Deleted Files

There are three steps in total:

- Tracing deleted files
- Find the path for the deleted files
- Go to that path using vfs(virtual File System)

### 5.2.3 Used Artifacts

Used artifacts for this feature are:

- **Windows.NTFS.MFT**
  Windows.NTFS.MFT refers to the Master File Table in the NTFS (New Technology File System) used by Windows. It serves as a crucial metadata repository, storing information about each file and directory on a disk. The MFT contains records with details such as file names, attributes, timestamps, and file permissions. Its efficient structure allows for quick access to file information, contributing to the overall performance of the file system. The MFT is a critical component in NTFS, enabling organized and rapid file retrieval and management.
- **Windows.NTFS.Recover**
  Windows.NTFS.Recover likely refers to a capability or artifact within the Windows NTFS (New Technology File System). It suggests a feature related to data recovery within the file system. In NTFS, recovery mechanisms may involve restoring deleted or corrupted files, directories, or recovering data from damaged sectors. This capability is crucial for file system maintenance and incident response, allowing users to recover lost or compromised data on NTFS-formatted drives.

### 5.2.4 Trace Deleted Files

**Steps:**

- Create a Hunt with "Windows.NTFS.MFT" artifact

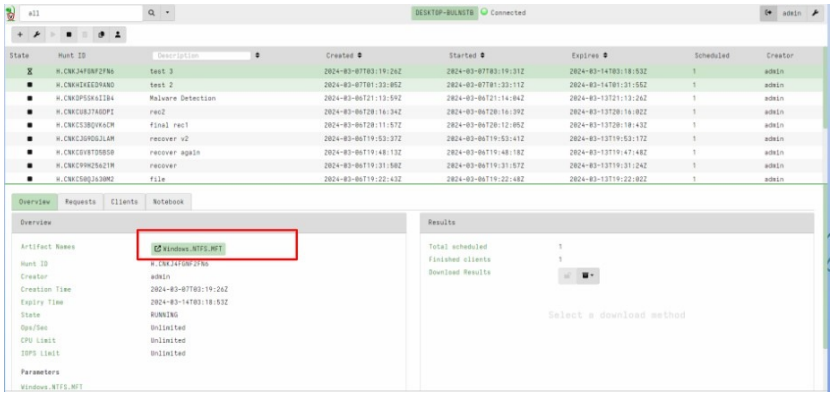


Figure 5.5: Trace Deleted Files-1

- Initiate the hunt and await its completion.
- After completing the hunt ,check for the result and desired file

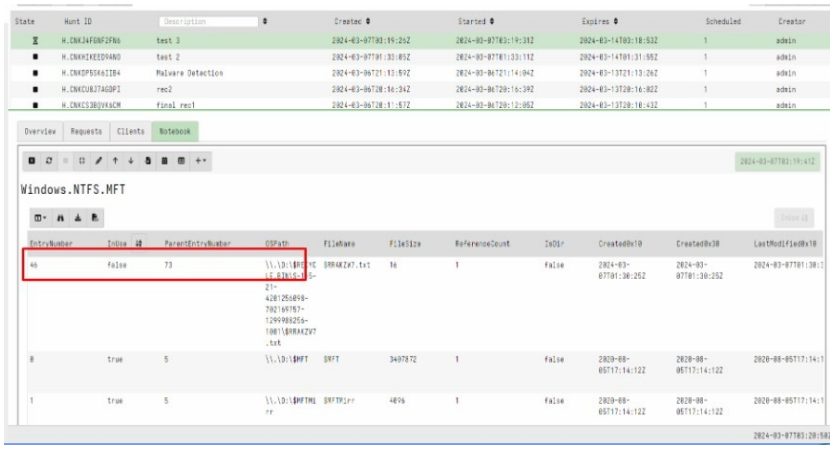


Figure 5.6: Trace Deleted Files-2

Here, we can identify the inode number of the desired file. Subsequently, execute another hunt using the artifact "Windows.NTFS.Recover" adjusting the parameters with the appropriate values.

### 5.2.5 Recover Deleted Files

**Steps:**

- Create a Hunt with "Windows.NTFS.Recover" artifact

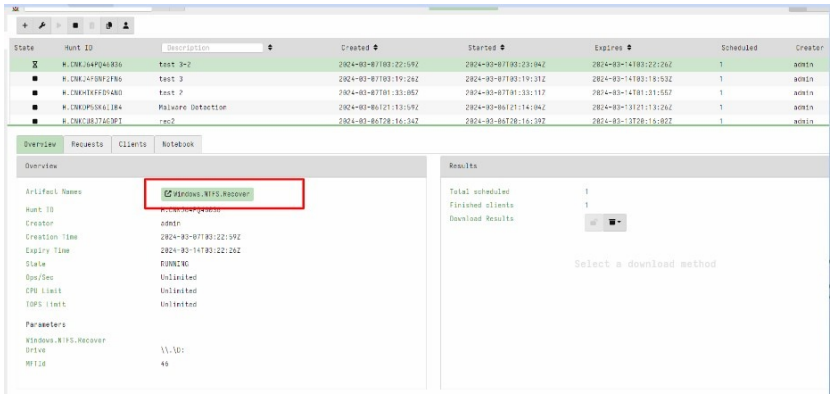


Figure 5.7: Recover Deleted Files-1

- Go to Notebook section and check the path

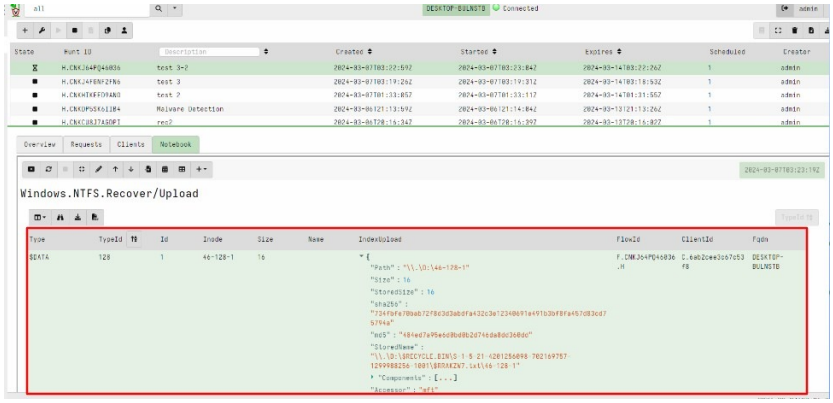


Figure 5.8: Recover Deleted Files-2

From here, we can find the past path and current path of the file after being deleted. If the inodes are replaced, then we can't recover the file.

### 5.2.6 Navigate to the file's path

**Steps:**

- Select client, go to it's Virtual File System

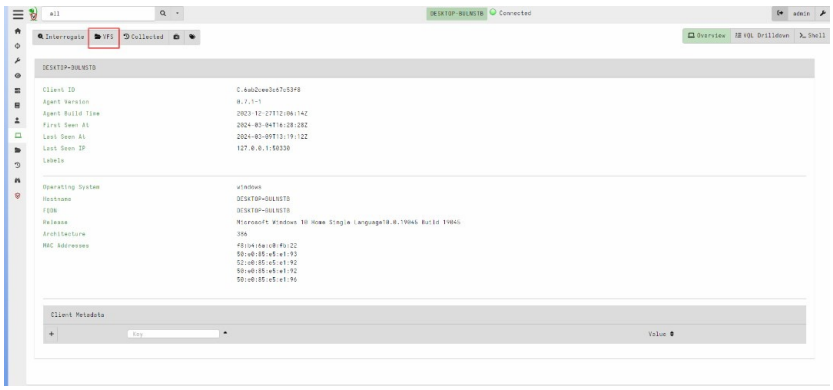


Figure 5.9: Find the File-1

- Navigate to the file's path and collect it

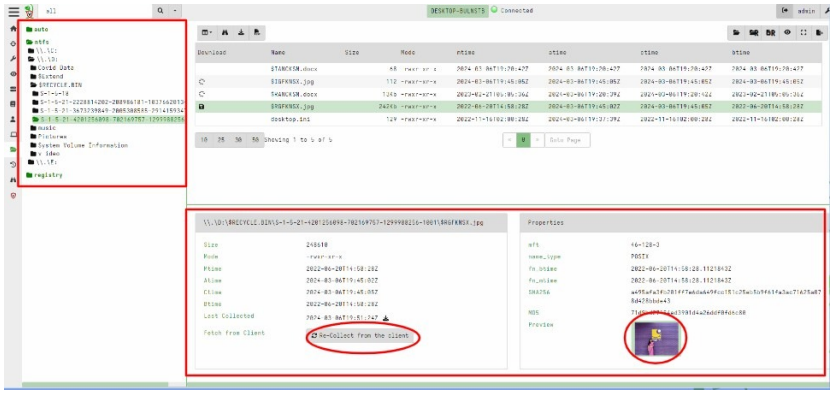



Figure 5.10: Find the File-2

In the event of inode replacement, the visible file will be the currently replaced one, not the previously deleted one.

## 5.3 Analyze Web Browser History

### 5.3.1 Why Analyzing Web History

In digital forensic investigation and cybersecurity analysis, web browser history is very crucial for the following reasons:

- Detecting Personal Behavior
- Inferring intent based on recently visited sites
- Indicator of Compromise(IoC) suggesting that the endpoint is susceptible to malicious attack
- Detecting Source of Infection for any malware or phising attacks
- Inferring Geolocation data

### 5.3.2 Used Artifact

**Windows.Application.NirsoftBrowserViewer** is used for this feature
NirsoftBrowserViewer is not a built-in support for velociraptor, rather it is a third party tool that velociraptor uses to fetch browser history data supported for the browsers like - Google Chrome, Safari, Internet Explorer, Mozilla Firefox, Microsoft Edge etc.

### 5.3.3 Get Web Brower History

**Steps:**

- First, we will create a hunt selecting the artifact mentioned above.
- Then, we will run the hunt. While the hunt is running, if we press the Download Results options shown in the following picture, then a .zip file will appear showing the size of the file. We will download it and unzip it.

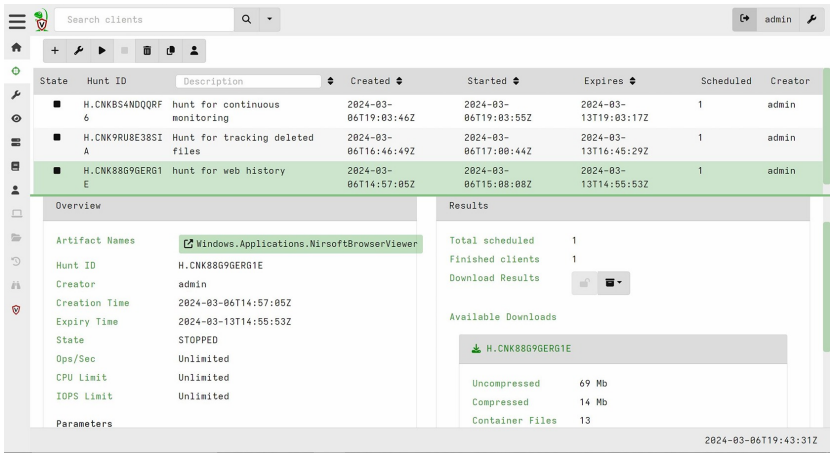



Figure 5.11: Get Web Brower History-1

- If we unzip the .zip file downloaded in step-2. We will see different folders starting with the term 'DESKTOP' and then an id. The id is represents a client. For each client, there will be a folder which will have a results folder, client information json file and an uploads file. Now, we move our focus to the results folder. If we open it, there will be 2 files -
  - Windows.Application.NirsoftBrowserViewer.csv
  - Windows.Application.NirsoftBrowserViewer.json

- For post-analysis of the .csv file in the results folder, we have written a python script that filters out the 'https' protocol and then extracts the domain names.

```python
import pandas as pd

# dataframe loaded
df = pd.read_csv('input.csv')

# filtered domains starting with https
filtered_df = df[df['URL'].str.startswith('https')]

# further filtering for domain detection
filtered_df['Domain'] = filtered_df['URL'].apply(lambda x: x.split("//")[1].split("/")[0])

# counting domains and sorting in descending order
domain_counts = filtered_df['Domain'].value_counts().sort_values(ascending=False)
print(domain_counts)
# Saving in CSV
# print("Writing in CSV files")
domain_counts.to_csv('sorted_domain_counts.csv')
```

Figure 5.12: Get Web Brower History-2

After that, the domain names are sorted based on their number of visit counts and the most visited sites are found which can be analyzed to understand the behavior of targeted endpoint.

- Check the result after running the script

| | A | B |
|---|---|---|
| 1 | **Domain** | **Count** |
| 2 | codeforces.com | 3474 |
| 3 | accounts.google.com | 2428 |
| 4 | www.google.com | 2154 |
| 5 | github.com | 1899 |
| 6 | mail.google.com | 1674 |
| 7 | docs.google.com | 1550 |
| 8 | www.figma.com | 1367 |
| 9 | supabase.com | 1247 |
| 10 | chat.openai.com | 1032 |
| 11 | moodle.cse.buet.ac.bd | 1001 |
| 12 | www.youtube.com | 713 |

sorted_domain_counts

Figure 5.13: Get Web Brower History-3

- We have also written a script which detects the most recently visited sites, which can be used to understand the recent behavior or intent of the endpoint.

sort_history.py 1

codes > sort_history.py > ...

```python
import pandas as pd

# dataframe loaded
df = pd.read_csv('input.csv')

# Visited column refers to last visiting time
df['Visited'] = pd.to_datetime(df['Visited'])

# We want the newest first
df_sorted = df.sort_values(by='Visited', ascending=False)

# Listing according to mostly newer websites first
df_sorted.to_csv('sorted_browsing_history.csv', index=False)
```

Figure 5.14: Get Web Brower History-4

The post-analysis codes are available on the link:
https://github.com/f12-mou/velociraptor_analysis.git

## 5.4 Continuously Monitor and Detect Unsuccessful Logins

### 5.4.1 Why Monitoring Logins

Unsuccessful login attempts can be an early indicator of unauthorized access attempts. Continuous monitoring can help detect these attempts in real-time which leads to quick response to potential breaches or attacks.

### 5.4.2 Used Artifact

**Windows.System.FailedLoginAttempts** is used for this feature
It is a custom artifact which we wrote to investigate the securityLogFile in windows and detect event id 4625. Event id 4625 is generated on the system where login attempt was made and it is failed, both for local and remote.

### 5.4.3 Detect Unsuccessful Logins

**Steps:**

- First, we have written a custom artifact which we named Windows.System.FailedLoginAttempts to trace the event id 4625 and raise an alert. The detailed code is given in the following Github link.

Windows.System.FailedLoginAttempts

Type: client

Custom Artifact

Parameters

| Name | Type | Default | Description |
|---|---|---|---|
| securityLogFile | | C:/Windows/System32/Winevt/Logs/Security.evtx | |

Source

```
1
2 SELECT EventData as FailedEventData,
3     System as FailedSystem
4 FROM watch_evtx(filename=securityLogFile)
5 WHERE System.EventID.Value = 4625
6
```

Figure 5.15: Detect Unsuccessful Logins-1

- Then we run the hunt selecting the artifact.
- While the hunt is running, we made an unsuccessful login attempt to detect whether the artifact is capable to detect the attempt real-time. We used here "runas" command in windows. The command was introduced first in Windows 2000. The command is used to create a process with alternate credentials. In the following picture, we want to run notepad. We willingly gave wrong password to simulate unsuccessful login attempt.
- Check the result after running the script

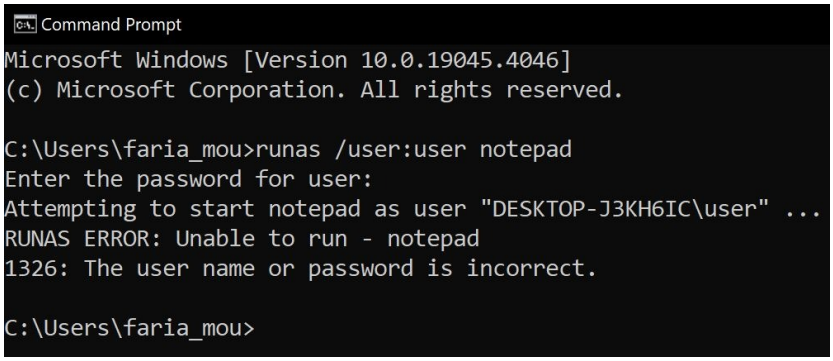



Figure 5.16: Detect Unsuccessful Logins-2

- If we switch to the 'Notebook' tab, then we can view the details of the unsuccessful login attempt.

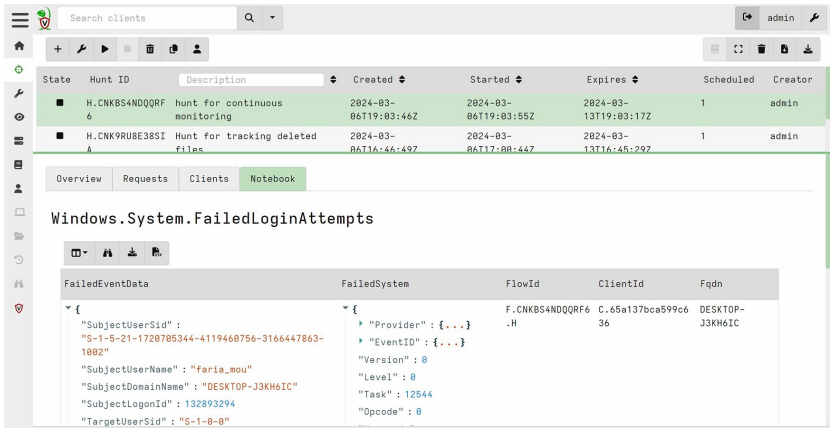



Figure 5.17: Detect Unsuccessful Logins-3

As we can see, a detailed real-time detection is here. The event object contains various information like SubjectUserSid, SubjectUserName, SubjectDomainName, SubjectLogonId, TargetUserSid etc. The fqdn tag is a reference for the desktop id of the endpoint.

Thus we can detect any unsuccessful login attempt real-time using the custom artifact.

The post-analysis codes are available on the link:
https://github.com/f12-mou/velociraptor_analysis.git

# Chapter 6

# Conclusion

## 6.1 Conclusion

In conclusion, Velociraptor stands out as a robust and versatile tool for digital forensics and incident response. With its intuitive interface and extensive features, Velociraptor proves to be an excellent choice for organizations seeking efficient and comprehensive endpoint visibility. Its customizable and automated capabilities make it a valuable asset for assessing and enhancing an organization's overall cybersecurity posture.

While Velociraptor excels in providing deep insights into endpoint activities, it does require a certain level of technical expertise for optimal setup and configuration. Despite this consideration, Velociraptor remains a powerful solution for proactive threat detection, offering the means to conduct in-depth investigations and respond swiftly to security incidents.

Future research exploring Velociraptor's effectiveness across various industries and contexts could yield valuable insights into its broader applications and potential advancements. Overall, Velociraptor emerges as a pivotal tool, empowering organizations to fortify their defenses against the evolving landscape of cybersecurity threats, thereby contributing to a more resilient and secure digital environment.