THREAT HUNT PLAYBOOK

WRITTEN IN COLLABORATION BETWEEN LATVIA AND CANADA
BY CERT.LV & CAFCYBERCOM

THREAT HUNT PLAYBOOK

Written in collaboration between
LATVIA and CANADA
by
CERT.LV
Canadian Armed Forces Cyber Command

Mr. Varis Teivāns
Technical Director
CERT.LV

Colonel Jason Smith
Chief of Staff
CAFCYBERCOM

18 Mar 2026

Threat Hunt Playbook

This publication provides a methodology for conducting threat hunting on critical information systems providing a repeatable systematic approach to identify compromises that were otherwise not detected using other security mechanisms. Developed as a collaborative effort between the Cyber Incident Response Institution (CERT.LV) and the Canadian Armed Forces Cyber Command (CAF), it documents various best practices gathered during active threat hunting operations in a number of institutions.

Joint threat hunting activities began in 2022 after the Russian invasion of Ukraine with an aim to find and remove any detected threat actors present in critical infrastrucure networks and improve the security posture of the institutions. The means by which this is conducted are captured in this playbook.

Threat hunting is one of many information security activities that an organization can employ to `Detect` and `Respond` to cybersecurity compromises and must be employed along side the other activities covering the other security functions for an effective cybersecurity management operation.

NOTE: The NIST Cybersecurity framework defines 6 security functions for managing cybersecurity risks: `Identify`, `Protect`, `Detect`, `Respond`, `Recover` and `Governance`.

Audience

- **Leadership and decision-makers** to understand objectives, decision points, deliverables, and how threat hunting improves defensive posture.
- **Threat hunters, analysts, and responders** for practical execution support, analytic approaches, and query examples.

Supplemental Content

This publication is continously updated and is supplemented with an annex that holds custom built Velociraptor Artifacts to support some of the activities described in the playbook.

Threat hunting definition

Although there are many definitions for Threat Hunting, we follow the National Institute of Standards and Technology (NIST) definition. NIST defines it as follows in NIST 800-53 RA-10 "Capability to Search for indicators of compromise in organizational systems; and to Detect, track, and disrupt threats that evade existing controls."

Threat Hunt Playbook

Table of Contents

Abstract

The Threat Hunting playbook provides a methodology and best practices for conducting threat hunting operations.

Developed as a collaborative effort between the Cyber Incident Response Institution of Latvia (CERT.LV) and the Canadian Armed Forces Cyber Command (CAF CYBERCOM), the playbook documents various best practices gathered during active threat hunting operations.

Joint threat hunting activities began in 2022 after the Russian invasion of Ukraine with an aim to find and remove any detected threat actors present in critical infrastructure networks and improve the security posture of the institutions. The means by which this is conducted are captured in this playbook. It is designed to share various methods for assessing the security posture of IT networks with an ephasis on finding compromised machines.

Introduction

Aim

This document aims to provide comprehensive guidance on conducting Threat Hunting (TH) operations in IT networks. It is intended to be continuously updated with additional procedures, serving as a living document to capture and communicate corporate knowledge at the tactical level of threat hunting operations. While there is a strong emphasis on specific tools such as Velociraptor, the playbook presents a foundational workflow that can be adapted for use with any tool suite.

The document includes:

- Background Information of various methods for conducting threat hunting.
- Standard Operating Procedure (SOP) to guide readers through efficient TH operations.
- Good Practices and Common Pitfalls for successful threat hunting and recommendations on issues to avoid.
- Example Hunts and Procedures to illustrate threat hunting techniques.
- Annexes: Example documents or tailorable queries to serve as a working reference, capturing specific techniques based on available tools.

Audience

This playbook is written for a mixed audience:

- **Leadership and decision-makers** to understand objectives, decision points, deliverables, and how threat hunting improves defensive posture.
- **Threat hunters, analysts, and responders** for practical execution support, analytic approaches, and query examples.

Scope

The scope of this playbook focuses on the hands-on, day-to-day work of TH analysts. It does **not** cover broader aspects of threat hunting operations, such as:

- Identifying authorities responsible for granting permission to conduct TH operations.
- Coordinating with partners involved in a TH operation.
- Establishing onboarding mechanisms for the chosen toolset on a client’s network to support TH activities.

Threat Hunting

Definition

Threat hunting operations are intelligence-driven and informed defensive cyber operations. These operations systematically search a network to identify and remediate threats that the existing security architecture has not detected or isolated. Unlike passive monitoring that waits for alerts, threat hunting actively seeks evidence of adversary activity through deliberate investigation, hypothesis testing, and pattern analysis.

Indicator based vs Tactics, Techniques and Procedures (TTP) based

> 📝 **Note** 📝
> This document uses *Indicators* and *Indicators of Compromise* (IoC) interchangeably.

Threat hunting can be conducted using various approaches, but the two most commonly used and effective methods are Indicator-Based Hunting and Threats, Tacticsm and Procedures (TTP) based Hunting. Each has its own advantages and disadvantages. The diagram below Figure 1 illustrates the relative difficulty of detecting malicious activity using these methods:

Indicator-Based Threat Hunting is easier to perform but heavily depends on the availability of high-quality threat intelligence. Analysts search for specific indicators such as IP addresses, domain names, and hash values. While this approach simplifies detection, its effectiveness relies on a reliable and up-to-date source of information.

TTP-Based Threat Hunting focuses on identifying host artifacts, tools, techniques, and procedures associated with threat actors. It is more complex and requires greater analytical expertise. Threat actors often exploit legitimate administrative tools to carry out malicious activities, a tactic commonly referred to as "living off the land." In these cases, skilled analysts must differentiate between legitimate and malicious activity.

Since indicator-based hunting is relatively straightforward, this document primarily emphasizes the processes and evidence required for TTP-Based Hunting. Analysts will learn how to confirm, detect, or disprove the presence of an attacker on a system.

> ⚠ **DISCLAIMER** ⚠
> While hunting for IoCs can be trivial, it is crucial to respect the infrastructure's available resources, particularly concerning file hashing and memory usage. Overloading systems with unnecessary demands may degrade performance and trust.

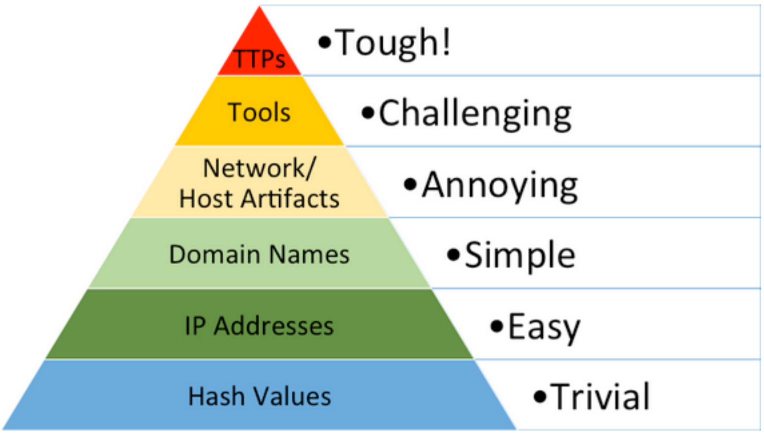


Figure 1 - Threat Hunting Methodologies

-
-

General Process

Threat Hunt Analysts must adhere to three main stages when conducting operations on unfamiliar infrastructure: Mapping, Hunting, and Posture Analysis. These stages must be executed in order, as mapping provides a foundational understanding of the client's tools, systems, and logging capabilities. This ensures a targeted and efficient hunt. Skipping this critical analysis can lead to slower or incomplete threat hunting due to overlooked details.

Stage 1: Infrastructure Mapping - Baseline

This stage involves analyzing and documenting the infrastructure to establish a baseline for the threat hunt. Key areas to examine include:

- **Operating Systems**: What types of operating systems are in use? For example, Windows, Linux, IoT devices, or mobile platforms.
- **Logs**: What logging policies are in place? What logs are available for analysis?
- **Internal Security Appliances**: How are security appliances utilized? Are they accessible to the threat hunting team? These devices often provide critical insights.
- **Security Posture**: What security procedures and measures are implemented? Examples include:
    - Second Factor Authentication (2FA)
    - Virtual Private Networks (VPNs)
    - Login policy
    - Firewalls
    - Previous security audits

Stage 2: Hunting

This stage involves actively searching for evidence of malicious activity across the infrastructure. Focus areas include:

- **Persistence**: Are there indications of activities designed to maintain persistence? Examples include:
    - Autoruns
    - Reg Keys
    - Scheduled Tasks
    - Cronjobs
- **Initial Access**: Is there evidence of suspicious remote access?
- **Lateral movement**: Is there evidence of malicious movement within the network? Examples include:
    - Suspicious named pipes
    - Workstation to workstation connections
    - Server to workstation connections
- **Exfiltration**: Is there evidence of data exfiltration? Indicators include:
    - High volume of outbound network activity
    - Compression of large amounts of data

Stage 3: Posture Analysis

This stage includes evaluating and enhancing the organization's overall security posture by analyzing and recommending improvements to configurations, policies, and defenses. Key areas to assess:

- SIEM (Security Information and Event Management) Configuration
- Group Policy Objects (GPOs)
- Antivirus Solutions (AV)
- Principles of Least Privilege and Least Functionality
- Software Restriction Policies (SRP)
- Lockout Policies
- Password Policies

- Multi-Factor Authentication (MFA)
- Firewall Configuration
- Cryptography Protocol Implementation

By adhering to these stages, threat hunt analysts can ensure thorough investigations, uncover hidden threats, and deliver actionable recommendations to enhance security.

Stage 1: Infrastructure Mapping

Pre-start

When preparing for a threat hunting operation, it is highly beneficial to obtain a detailed description of the client's infrastructure security posture. This can include documents such as network maps, Group Policy Objects (GPOs), and lists of devices with their respective operating systems. If such documents are unavailable, a comprehensive questionnaire should be provided. The questionnaire should address the following points:

- What operating systems are in use?
- What devices are deployed?
- Are logs being collected? If so, how long are they retained?
- Does the institution have its own Security Information and Event Management (SIEM) system?
- What antivirus solutions are in use?
- What software is approved for use?
- Is there a software restriction policy in place?
- Is a Virtual Private Network (VPN) being used?
- What is the institution’s Two-Factor Authentication (2FA) policy?
- What is the institution's public-facing IP space (e.g., IP ranges exposed to the internet)?
- Any additional infrastructure-specific questions.

These questions help threat hunters develop a clear understanding of the target infrastructure, including details about operational technology (OT) and IT devices, categorized by operating systems. This knowledge is crucial for understanding potential lateral movement options for attackers. Additionally, questions regarding software provide insights into whether observed software usage is authorized or anomalous. The goal of this phase is to provide context to the threat hunters as they begin their investigation and encounter potentially suspicious activities.

An example questionnaire can be found in Annex B.

Post-start

While pre-start information is valuable, experience shows that it is often inaccurate to varying degrees. For example, network maps, if available, may contain outdated or incomplete information, including non-existent endpoints or missing critical devices.

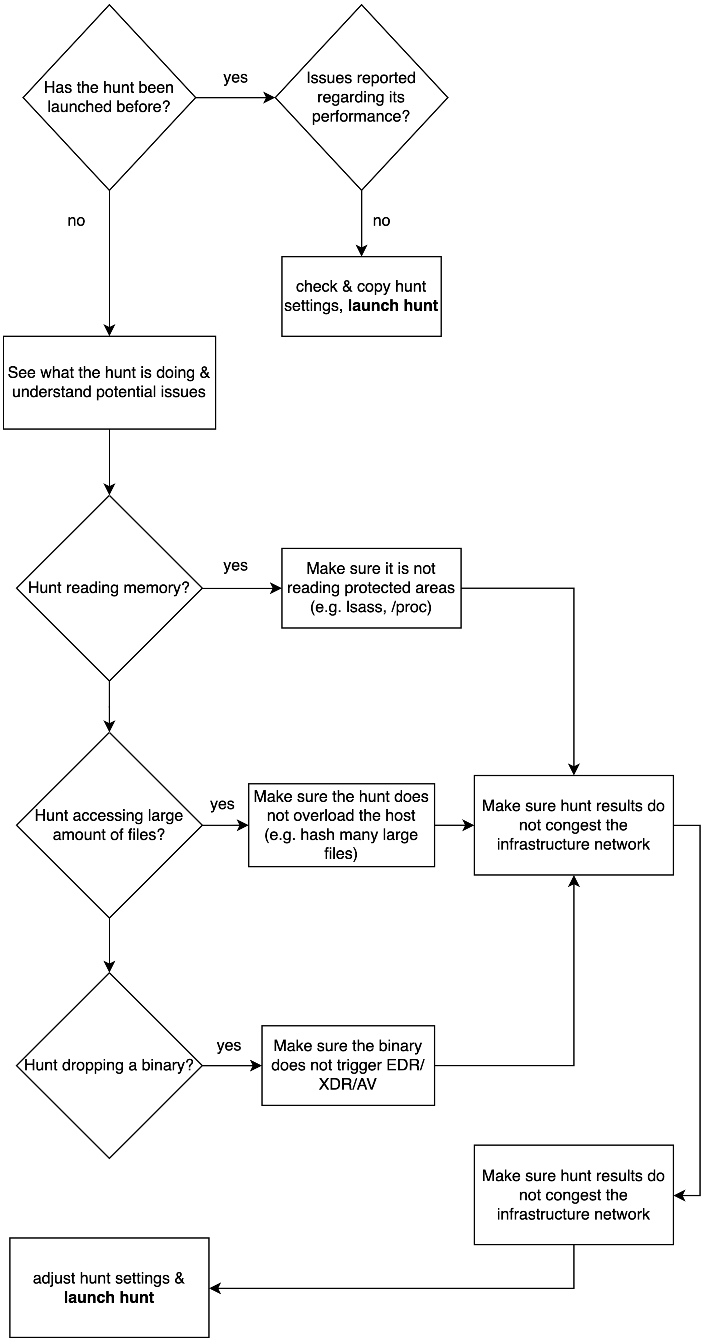
To address this, threat hunters must gain a practical understanding of the network using dedicated tools. This helps develop a baseline or “feel” for the network, encompassing typical processes (e.g., Endpoint Detection and Response (EDR) / Extended Detection and Response (XDR), SIEM, antivirus, VPN usage), connections, and programs running across systems.

Such baseline documentation should ideally be established during the client onboarding phase. While documentation is essential, active hunting should always take precedence. An example baseline document is provided in Annex C.

Stage 2 : Hunting

Resource Management

During the hunting phase, it is crucial to manage resource consumption carefully to avoid overburdening endpoints and the target network. Factors such as high resource utilization, network congestion, and potential reactions from XDR systems must be considered. Before launching new hunts, assess the potential impact, as shown in the accompanying thought-process diagram.


Has the hunt been launched before?
yes
Issues reported regarding its performance?
no
no
check & copy hunt settings, **launch hunt**
See what the hunt is doing & understand potential issues
Hunt reading memory?
yes
Make sure it is not reading protected areas (e.g. lsass, /proc)
Hunt accessing large amount of files?
yes
Make sure the hunt does not overload the host (e.g. hash many large files)
Make sure hunt results do not congest the infrastructure network
Hunt dropping a binary?
yes
Make sure the binary does not trigger EDR/ XDR/AV
Make sure hunt results do not congest the infrastructure network
adjust hunt settings & **launch hunt**

Test runs should be conducted if there is a risk of operational disruption. If the target institution uses EDR on endpoints, communicate with the administrators to monitor dashboards and report observations during test hunts. Running test hunts locally in a lab environment that mimics the target network is highly recommended.

Persistence

Once an attacker has gained access to a system, they typically aim to ensure that their access is not lost during a reboot, and thus, they establish *persistence*. This step is crucial for the attacker to maintain long-term control over the system. There are various ways to achieve persistence, with one of the most common methods being the use of scheduled tasks or actions, which can trigger the execution of malicious code at specific intervals or events.

Persistence techniques vary depending on the system in question, with different approaches being used on different systems systems.

Windows

There are several methods to establish persistence on a Windows machine, all of which should be thoroughly examined during a threat hunt. Below is a brief overview of key areas to investigate.

- **Registry Run Keys**: Run keys can be modified to execute a malicious file every time a user logs on (via the Run key) or just once after login (via the RunOnce key), after which the key is deleted.
    - `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`
    - `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce`
    - `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run`
    - `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce`

- **Services**: Windows services are processes that run in the system background without requiring user interaction. Since services typically start automatically during system boot, they are attractive targets for attackers seeking persistence. Attackers can create new services or modify existing ones to ensure that their malicious code or program is executed every time the system starts.

- **Scheduled Tasks / Autoruns**: Mechanisms to execute scripts or commands at specific times. Attackers can leverage these to execute malicious commands or scripts on a schedule.

> 📝 **Note** 📝
> Threat hunting tools often have specific hunts to investigate autoruns.

When examining Scheduled Tasks and Autoruns, the goal is to identify suspicious entries. Information gathered during reconnaissance about approved software on the infrastructure is critical here. If unauthorized software, also known as non-whitelisted software, is found in autoruns, it should be flagged as suspicious. A Request for Information (RFI) should then be sent to the infrastructure administrators to verify the legitimacy of the flagged entry.

> 💭 **Living Off The Land Binaries (LOLBins)** 💭
> Attackers can exploit LOLBins to execute malicious scripts. Searching for LOLBin usage in autoruns or scheduled tasks can be an effective way to uncover suspicious activity.

Linux

Methods of achieving persistence on a Linux machine differ from those on Windows. Linux systems, being highly customizable and configurable, offer attackers greater opportunities for hiding malicious code.

The following is not an exhaustive list but serves as an introductory guide with key insights and examples:

- **Startup Scripts**: Scripts such as /etc/rc.local, ~/.bashrc, and ~/.bash_profile can be modified to execute malicious code during system boot or when a user logs in.

- **Cron Jobs**: The Linux equivalent of Windows ***Scheduled Tasks / Autoruns***, cron jobs are highly popular among attackers due to their ease of management and customization.

- **Systemd Jobs**: These provide fine-grained control over services and run under the systemd suite. They can be configured to start or stop after specific events. The large number of systemd services on a typical Linux system offers attackers flexibility to hide and execute malicious scripts.

- **Hijacking library functions**: This technique involves modifying existing services and functions to add malicious functionality. For example, attackers can create a `.so` module to intercept legitimate runtime processes using the `LD_PRELOAD` variable or by modifying configuration/source files of startup applications. Malicious `.so` modules can perform tasks such as logging credentials or creating reverse shells.

- **Hijacking Linux kernel**: Operating in "ring 0," the Linux kernel interacts directly with hardware, enabling highly stealthy persistence techniques. Attackers can create a malicious driver, which is then loaded into the kernel using commands like `insmod` or `modprobe`. However, this method is more complex than creating cron jobs or systemd services, as it requires creating, compiling, and loading the driver's source code into the kernel.

Other Persistance Techniques

- **Web shells**: A web shell is malicious code uploaded to a web server to provide attackers with remote control of the device. Often used alongside network tunneling tools, web shells allow attackers to execute commands on the victim machine. Analysts can use YARA rules to scan files for patterns or characteristics associated with malicious code.

In Velociraptor there are several artifacts designated specifically for this task, for example `Generic.Detection.Yara.Glob`.

> 📝**Note** 📝
> The `UploadHits` option in Velociraptor artifacts should be used with caution. If a YARA rule triggers numerous false positives, enabling this option can overwhelm the target infrastructure's bandwidth by uploading all matching files to the Velociraptor server.

- **Tunneling**: Attackers may use VPNs or proxy services, initiated from persistence mechanisms, to establish encrypted communication channels with their Command & Control (C2) device.

Initial Access

Before an attacker gains a foothold into the system they are targeting, they must first achieve *Initial Access*, meaning they must compromise a device or system. There are numerous ways an attacker can gain this access, but they only need to succeed with one method. As defenders, it is crucial to proactively analyze the system's attack surface and identify its weakest points.

A key consideration is identifying vulnerable services or internet-facing systems that could be exploited through known vulnerabilities, such as Common Vulnerabilities and Exposures (CVEs). These services should be carefully reviewed to determine if they are running outdated or insecure versions. Any system that is exposed to the internet and vulnerable to multiple exploits should be considered at high risk, and potentially already compromised.

Authentication mechanisms are another critical area. Weak or poorly managed authentication elements - such as reused passwords or lack of multi-factor authentication - are prime targets for attackers. In the threat hunting process, it is essential to work closely with infrastructure administrators to understand the security measures in place and assess any weaknesses in the authentication processes.

Additionally, communication with system administrators is essential when reviewing remote administration tools. Many organizations use remote access solutions like Remote Desktop Protocol (RDP) or TeamViewer. Confirming with administrators which tools are legitimately part of the infrastructure is important to ensure that unauthorized tools are not being used by attackers to gain initial access. Examples of useful information to verify:

- Presence of internet-facing services.
- Availability of RDP or VPN logs.
- Successful authentications from unusual locations.

Lateral Movement

Compromising a single device is rarely sufficient for an attacker to establish a steady foothold within a target system. To achieve their objectives, attackers often move laterally within the network, leveraging or escalating permissions to access otherwise inaccessible parts of the environment.

Attackers can achieve lateral movement through various methods, often exploiting legitimate administration tools, including:

- Windows Management Instrumentation (WMI)

- Abuse of management tools
- PsExec

Lateral movement refers to the techniques employed by threat actors to expand their reach within a network after gaining initial access. Once inside, they aim to extend their influence, compromise additional systems, and escalate privileges.

Why looking for lateral movement is crucial:

- Lateral movement often precedes data exfiltration or other malicious actions, making it critical to disrupt this step in the attack chain.
- Identifying lateral movement allows for quick containment and response, limiting an attacker's ability to advance within the network.
- By recognizing lateral movement techniques, organizations can address vulnerabilities and implement additional safeguards.
- Analyzing lateral movement provides insights into the tactics, techniques, and procedures (TTPs) of the threat actor, aiding in attribution.

Indicators of lateral movement:

- PowerShell and RDP (internal remote execution)
- Privilege escalation (e.g., Users, DomainRoles)
- Pass-the-hash attacks
- Kerberoasting
- Living-off-the-land techniques
- Phishing (e.g., Browser history, antivirus quarantines)

Exfiltration

Exfiltration involves the unauthorized transfer of sensitive or confidential data from an internal network to an external location controlled by a threat actor. This stage typically occurs after the attacker has gained access, moved laterally, and identified valuable information.

Why detecting exfiltration is essential:

- Detecting exfiltration helps safeguard sensitive data and prevents unauthorized disclosure.
- Timely detection reduces financial, reputational, and legal consequences of a cyber incident.
- Many compliance standards mandate the protection of sensitive data, making exfiltration detection a critical requirement.
- Detecting and preventing exfiltration protects intellectual property, trade secrets, and other proprietary information from theft.

Indicators of Exfiltration:

- Command and Control traffic (e.g., DNS, HTTP, FTP)
- External medium usage (e.g., Registry keys)
- Web services used for data transfer
- Abnormally large data transfers
- Scheduled transfers (e.g., PowerShell scripts, Scheduled Tasks)

When attackers maintain access to a system over an extended period, they often begin to exfiltrate sensitive data, such as email exchanges, confidential project files, or other private information. Data exfiltration can occur through a variety of methods, which can make it difficult to detect and confirm. Identifying signs of exfiltration is critical in determining whether the compromise was limited to the system itself or part of a larger incident involving the exposure of sensitive or classified data.

Attackers typically leverage multiple tools and services to exfiltrate data. Monitoring the following areas can help in detecting such activity:

- File-Sharing Services - look for unauthorized tools. Network administrators should provide a list of approved services to assist in identifying anomalies.
- Cloud Sharing Services - flag any unexpected connections to cloud servers or domains.
- Microsoft Exchange Mailbox Export - watch for mailboxes being exported to remote Microsoft Exchange servers.

Stage 3 : Posture Analysis

Posture analysis is not a standalone stage but a continuous process throughout the threat hunt. It involves identifying and analyzing artifacts that indicate poor infrastructure administration practices and internet-facing vulnerabilities.

Infrastructure administrators may inadvertently weaken security through actions such as:

- Weak password policies
- Allowing remote management access to servers from external IPs
- Storing credentials in plain text files
- Using weak cryptographic protocols
- Maintaining a non-homogeneous tool environment
- Assigning identical passwords for local administrators across the network

These bad practices provide attackers with additional opportunities to move laterally, escalate privileges, and exfiltrate sensitive information such as emails or documents.

Vulnerable servers and applications present significant risks. Analysts should gather version information to search for potential exploits and correlate findings with device inventories for deeper analysis. The public IP ranges of the infrastructure should be researched using open-source tools like Shodan or Censys. A report detailing vulnerable operating systems and applications should be created and used for further investigation.

Threat Hunting Methodology

Grouping and Counts

When searching for anomalies it is recommended to sort the data by the number of occurrences. Strong SOPs in place this will it easier to occurrences of abnormal behavior. The same applies if an activity is frequently occurring across the network, as this may indicate either a planned activity or a heavily compromised network.

Persistence Vs Initial Access Vs Lateral Movement

Assuming that the network under analysis is already compromised allows an easier and repeatable way to approach new networks. If persistence is found the task can be escalated to Lateral Movement and Initial Access concurrently.

Most threat actors intend to avoid their presence and persistence being detected for as long as possible. To accomplish this, they tend to reduce the amount of noise their activities generate on the network. This can translate in establishing persistence on the least number of hosts possible while still maintaining freedom of maneuvers. Following this thought process, detection techniques could involve analysis of low occurrences of scheduled tasks and autoruns that have run across the enterprise.

If Initial Access is considered the starting point, further investigation is needed on all potential leads to identify any persistence associated with them. This will create an insurmountable task of cataloging not only the vulnerable software but also the various ways that it can be exploited.

Lateral Movement, if used as the starting point, requires a strong knowledge of not only how the network is used but also how the users accomplish their daily tasks. This changes drastically from network to network and user to user.

Therefore this is not a repeatable process from network to network.

Detection of Exfiltration

If proof of exfiltration is discovered on the network, it is often the case that persistence is already in place. While it is possible that a Threat actor's objective is to exfiltrate data from a specific host, they would most likely implement persistence and C2 capabilities to expand their exfiltration efforts further considering the initial effort deployed to gain this access.

Tools

Threat Hunting can be done via multiple tools. There is no *one* tool that is best for performing Threat Hunting operations, as there exists a multitude of EDR solutions and other software that can be leveraged to find threats in a target infrastructure.

Not all institutions and companies have the resources to invest in these solutions. This is why open-source solutions with the goal of achieving consistency accross our threat hunting operations and eliminating the need for the analyst to learn a new EDR/SIEM or other solution every time a new theat hunting operation begins.

CERT.LV and Canadian Cyber Task Force 2 primarily leverage Velociraptor (*Mike Cohen - Rapid7*) and Elastic (*Elastic NV*).

Velociraptor provides full access to the host machines and allows the performance of advanced digital forensics, search for indicators and find patterns of malicious activity. While Elastic Fleet enables the leveraging of logs in order to add depth to findings.

Velociraptor

Velociraptor is an advanced digital forensic and incident response (DFIR) tool that used to perform threat hunting operations. It is built up from three fundamental blocks - the Velociraptor Query Language (VQL), artifacts, and hunts.

VQL

VQL uses functions and plugins to create queries which manipulate data and implement logic. It allows for the execution of complex tasks ranging anywhere from collecting log files to downloading, running, and executing external tools (e.g. yara, autoruns and more).

Artifacts

An artifact is a package that consists of one or more VQL queries in the format of a human readable YAML file with a custom name, the goal or task of the artifact can often be derived from its name. An artifact file simply embodies the query required to collect data or answer a specific question about the endpoint.

> **📝Note📝**
> When importing artifact packs into Velociraptor, older versions only accept .yaml files, whereas newer versions also support .yml files.

There are several useful artifacts that come with the default Velociraptor setup, but it is suggested to also add the artifact exchange package - it consists of artifacts contributed and shared by the community. To automatically import the entire content of the artifact exchange into a server, run the Server.Import.ArtifactExchange artifact.

Reference: https://docs.velociraptor.app/exchange/

A collection of artifacts can return rows or upload files back to the velociraptor server. This is because an artifact is simply a VQL query and all queries return a sequence of rows. These results can be viewed & downloaded by an analyst on the Velociraptor server frontend.

Reference: https://docs.velociraptor.app/docs/gui/artifacts/

Hunts

A hunt is a collection of a one or more artifacts from a group of systems, i.e. multiple endpoints connected to the Velociraptor server execute one or more artifacts, gather the data, and send it back to the server for analysis.

Reference: https://docs.velociraptor.app/docs/gui/hunting/

- Analyzing hunt results
    - Statistical analysis of autoruns, amcache

Flows

At its core a flow refers to a chain of events executed by a client or the server. Whenever an artifact is executed, a flow is created and it can be inspected to see the details of the artifact execution and data collection. For example, if a hunt with two artifacts is launched on a network of 10 hosts, it generates 20 flows—one flow for each artifact on each client.

The application stack available to the analyst is designed for rapid & detailed data analysis, processing and reporting with the main hunting tools being Velociraptor and Elastic.

Before delving into details on how Velociraptor can be leveraged for threat hunting, one needs to understand a few basics of the Velociraptor Query Language. A good Velociraptor overview and introduction to the Velociraptor Query Language (VQL) fundamentals can be found in this webinar (available online at: https://youtube.com/playlist?list=PLz4xB83Y3Vbjtqr_ttOkBWZZ_ewEVVPXQ).

VQL Commands

This section details some basic VQL functionality. Note that more extensive documentation can be found in https://docs.velociraptor.app/. Velociraptor developers have also published detailed educational materials on https://training.velociraptor.app/.

Selecting Results to Display

SELECT * FROM = Pull all available columns as part of the query

SELECT Column1, Column2, Column3 FROM = Pull specific columns when looking for particular data. This approach can reduce query time and declutter the displayed results. Ensure that the column names match exactly what is found in the results.

SELECT `ColumnName WithSpace` FROM = If column names contain spaces or special characters, use backticks (`) to reference them.

Grouping and Ordering

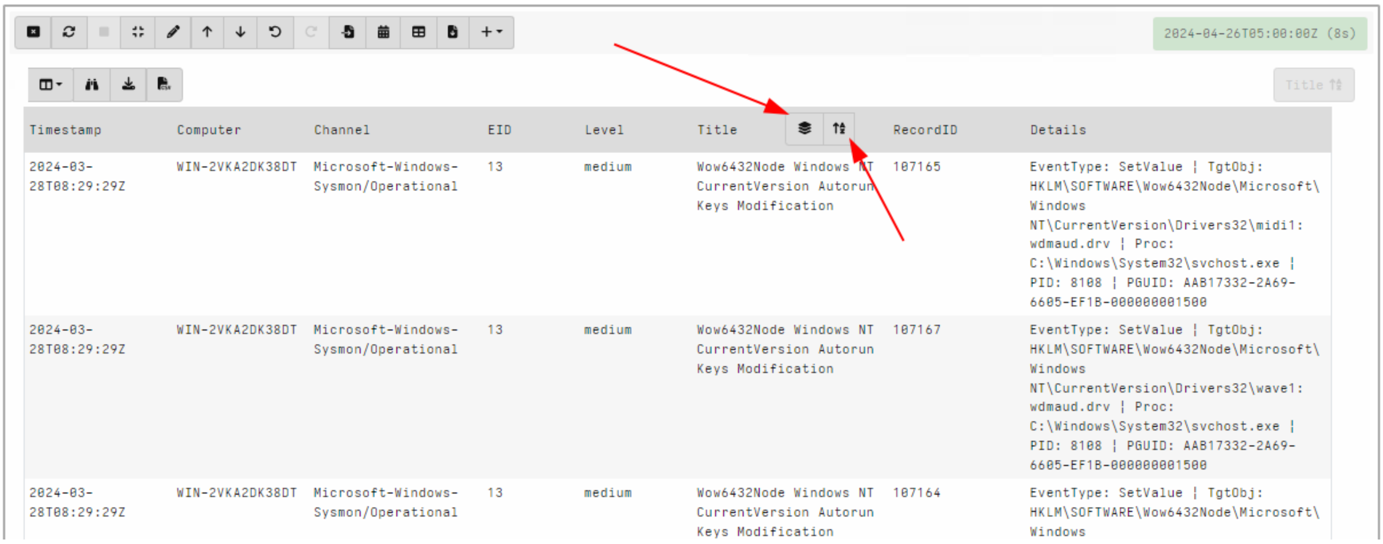
GROUP BY ColumnOfYourChoice = It will group results based on a chosen column. Using a count() column will also show the number of occurrences for each result following the GROUP BY.

ORDER BY Count DESC = Results will be sorted based on a chosen column. In this case, the Count column will be sorted in descending order. If "DESC" is not specified, it will default to ascending order.

GUI Based Stacking

The issue with the above approach is the speed and the cumbersome workflow. If we wanted to change our query, we would have to issue another 'GROUP BY' clause which takes time and impedes the analysis process.

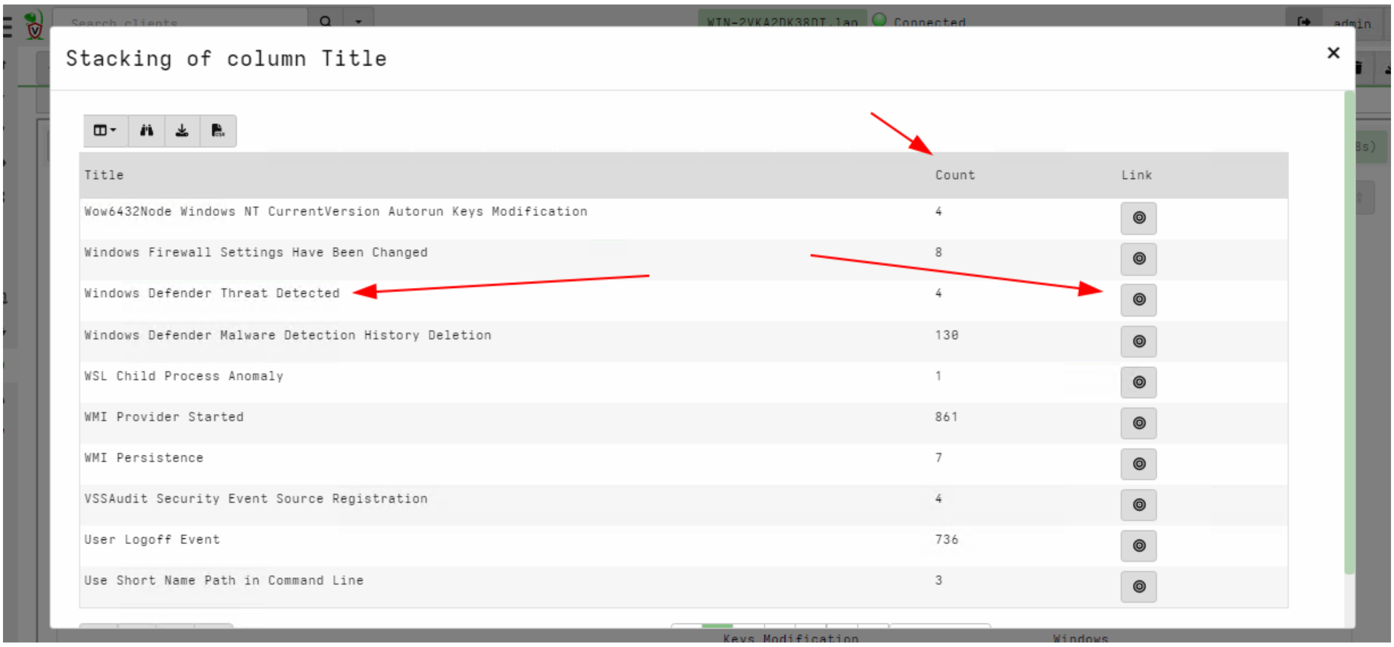
In Velociraptor version 0.72, stacking support is now built in to the GUI. Once a column is sorted, a 'stacking' icon will appear beside it. Clicking on this icon will open a new stacking window showing the total counts of various rows, and a 'Line' column with a button that allows the hunter to quickly view results from each specific row type.



2024-04-26T05:00:00Z (8s)

Timestamp	Computer	Channel	EID	Level	Title	RecordID	Details
2024-03-28T08:29:29Z	WIN-2VKA2DK38DT	Microsoft-Windows-Sysmon/Operational	13	medium	Wow6432Node Windows NT CurrentVersion Autorun Keys Modification	107165	EventType: SetValue ¦ TgtObj: HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Drivers32\midi1: wdmaud.drv ¦ Proc: C:\Windows\System32\svchost.exe ¦ PID: 8108 ¦ PGUID: AAB17332-2A69-6605-EF1B-000000001500
2024-03-28T08:29:29Z	WIN-2VKA2DK38DT	Microsoft-Windows-Sysmon/Operational	13	medium	Wow6432Node Windows NT CurrentVersion Autorun Keys Modification	107167	EventType: SetValue ¦ TgtObj: HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Drivers32\wave1: wdmaud.drv ¦ Proc: C:\Windows\System32\svchost.exe ¦ PID: 8108 ¦ PGUID: AAB17332-2A69-6605-EF1B-000000001500
2024-03-28T08:29:29Z	WIN-2VKA2DK38DT	Microsoft-Windows-Sysmon/Operational	13	medium	Wow6432Node Windows NT CurrentVersion Autorun Keys Modification	107164	EventType: SetValue ¦ TgtObj: HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows

Figure 3 Velociraptor 0.72 Stacking Button



Stacking of column Title

Title	Count	Link
Wow6432Node Windows NT CurrentVersion Autorun Keys Modification	4	◎
Windows Firewall Settings Have Been Changed	8	◎
Windows Defender Threat Detected	4	◎
Windows Defender Malware Detection History Deletion	130	◎
WSL Child Process Anomaly	1	◎
WMI Provider Started	861	◎
WMI Persistence	7	◎
VSSAudit Security Event Source Registration	4	◎
User Logoff Event	736	◎
Use Short Name Path in Command Line	3	◎

Figure 4 Velociraptor 0.72 Stacking Window

> 📝**Note**📝
> This requires Velociraptor version 0.72 or later.

Search for Specific String

AND NOT Column3 =~ "Microsoft" = Excludes values from results with the NOT parameters. This query will exclude any row in Column3 that contains "Microsoft".

=~ "Micro" VS **= "Micro"** = This first example is VQL's version of regex lookup. Most syntaxes for regex are available. In this example the query will look for any value that contains "Micro". The second example is a literal lookup, where it will only look for column values that match exactly "Micro".

Search Per Count

SELECT *, count() as Count FROM = This will add a new column named Count which is working as the count() function. Its purpose is to count how many times each row occurs. If your results are not grouped, the count will just give each row a number based on how it is sorted (1, 2, 3, 4, 5, 6....) and is not very useful.

WHERE Count <= 5 = This will return all rows with the Count column where the value is less than or equal to 5.

Formatting Field

SELECT *, timestamp(epoch='Column EpochTime') as Time FROM = If a time column is in EPOCH time, it can easily be converted to a human-readable format using timestamp().

SELECT *, hash.MD5 FROM = If columns are in JSON format, such as a hash column containing MD5, SHA1, and SHA256 in one cell, one can choose which one to display or assign them to their own individual columns.

Filtering Large Data Results

LET Variable1 = SELECT * FROM.. = Entire queries an be stored in a variable. The results stored in this variable can be reused without processing the same query multiple times. For example, Velociraptor has a 10 minutes maximum query processing time limit which can be optimized with the use of one or more variables.

Parallelize() = Can be used to speed up notebook queries a bit with some observable limitations.

- Loss of flowid, clientid and fqdn columns from dataset when using parallelize.
- parallelize() cannot be used inside a variable.

Finding & Using IoCs when Hunting for Advanced Persistant Threats

Research what Advanced Persistant Threats (APTs) are likely active in the current region. For example, it is known that Northern Europe is targeted by Cadet Blizzard (formerly tracked as DEV-0586) according to Microsoft. Threat hunting operators should create VQL statements to search for APT artifacts in the target organisation based on such reports. Several examples are described below.

Example - Searching for the scheduled task splservice and spl32 since it's known that these services were used to communicate with a command and control (C2) server. Using Windows.Sysinternals.Autoruns hunt and will filter for the mentioned services.

```
SELECT * FROM hunt_results(artifact = "Windows.Sysinternals.Autoruns", hunt_id = 'YOUR HUNT ID')
WHERE Category="Tasks"
AND (`Launch String` =~ "splservice" OR `Launch String` =~ "spl32")
--Enable the following line if there is a result from above
--AND (`Launch String` = '2\>\&1' OR `Launch String` = '127\.0\.0\.1')
```

Often it's useful to search for known IoCs like IP addresses, domains, SHA/MD5 hashes, file names etc., and then we can apply several Artifacts.

Hunting For Hashes

When hunting for indicators of compromise (IoCs) related to Advanced Persistent Threats (APTs), hash-based searches are a common method. However, there are some important considerations to keep in mind regarding the effectiveness and limitations of hunting for hashes.

In Velociraptor, whenever a new client is seen by the server, it automatically creates a local SQL database using the Generic.Forensic.LocalHashes.Init artifact. This database is populated with hashes from user-accessible locations on the system using the Generic.Forensic.LocalHashes.Glob artifact. This allows for fast enterprise-wide hash searches. The SQL database will only contain hashes for files located in directories where regular users have write permissions, such as C:\Users\user\Desktop.

However, this method has limitations:

- **User land only**: The SQL database focuses on user space and does not account for kernel-level hashes, potentially missing important indicators. This can lead to false negatives, where malicious activity occurring in kernel space is not detected.
- **Known IoCs**: Once IoCs, such as hashes, are publicly released, most antivirus tools will likely already have these hashes identified and flagged. If the hash corresponds to known malware, modern antivirus solutions typically catch it unless certain folder exclusions are in place. Therefore, unless you have access to intelligence that isn't widely known, simply hunting for these hashes can be less valuable.

Despite these limitations, hash-based searches can still be useful for detecting known malicious files, especially if they are newly deployed or haven't been widely flagged yet. Hash hunting can be done effectively using the Generic.Forensic.LocalHashes.Query artifact to specify a list of hashes that should be queried on the endpoints.

Here is an example of using Velociraptor for searching known file names associated with a specific APT group:

Example: Search for known file names used by the aforementioned APT.

```
Artifact - Windows.Search.FileFinder

SearchFilesGlobTable:
C:\**\{dump64,procdump64,procdump,GOST,NGROK,IVPN,impacket,dbus-
rpc,Teamviewer,SurfShark,Python,Tor,netcat,USORead}.exe
```

If further specificity is needed, you can also use Windows.Search.FileFinder or the VFS to manually search for specific hashes.

DNS Lookups

A crucial part of hunting for APT activity involves analyzing DNS traffic, as many APTs rely on domain names for command-and-control (C2) communication. Therefore, scanning DNS logs for suspicious or malicious domains is essential.

If the Velociraptor agent is deployed on a DNS server within the network, you can collect DNS logs for deeper analysis. Depending on the DNS server software in use, such as BIND, you may find it more efficient to directly search through DNS logs for known malicious domains using tools like grep, rather than ingesting large log files into Velociraptor.

However, if you prefer to use Velociraptor, you can create an artifact to remotely search and analyze DNS logs. For example, you could use the following VQL to search for specific malicious domains in DNS traffic:

Example: Searching for malicious domains in DNS logs.

```
SELECT *
FROM source(artifact="Windows.System.DNSCache")
WHERE Record =~ "fbi.fund"
OR Record =~ "njratcasanew.ueuo.com"
OR Record =~ "imolaoggi.eu"
OR Record =~ "nirsoft.me"
OR Record =~ "pastebin.pl"
OR Record =~ "skreatortemp.site"
OR Record =~ "winrarsolutions.com"
```

While this approach works for DNS cache data on Windows systems, when dealing with DNS logs from servers like BIND, it may be more efficient to directly run a grep command on logs such as named.log. You can do this by utilizing an execve call or an artifact that facilitates remote searching, allowing for a faster and more effective detection of malicious domain activity.

Key Hunts

Autoruns

- Velociraptor hunts artifacts
    - Windows OS hosts: Windows.Sysinternals.Autoruns
    - Linux OS hosts: Custom.Linux.System.Autoruns
- Why use it?
    - Autoruns help to identify programs, processes, services and drivers that automatically run when a computer is started and a user logs in.
    - Analyzing Autoruns is crucial for various reasons including to identify potential malware, persistence mechanisms, unique hashes and timestamps which are important evidence necessary to understand the enemy behaviors.

    - A large list of hashes can be collected and then analysed through a malware scanning service as VirusTotal.

- Autoruns filtering recommendations:
    - **Time:** Indicates the time when the executable was created, not when added in Autoruns entries. Which helps identify if the binary is old or recent.
    - **Hashes:**
        - Compare to LOLBins
        - Analyze through Malware analysis tool as VirusTotal (website or API)
    - **Enabled:**
        - Enabled: If enabled, they will be launched at every system startup.
        - Disabled: While disabled entries won't enable persistence for the threat actor, they might indicate malicious activities which occurred prior.
    - **Signer:**
        - Verified: Verified entries are most likely legitimate binaries and should not be the first priority. But it is still useful to analyze quickly if nothing else is found.
        - Not Verified: Entries which are not signed, while not automatically malicious, should be looked at more closely.
    - **Launch String:**
        - File path: Where the executable is located and launched from.
        - Additional commands: Extra parameters could be added on the executable as part of the autoruns entries
    - **Company:** It's possible to exclude known and trusted companies, such as Microsoft, if the volume of results is too high. The logic is that Microsoft binaries are usually signed and not malicious.
    - **Count:** Enable to analyze the Autoruns entries occurring the least across all hosts on the enterprise.

Powershell

- Velociraptor hunts artifacts
    - Windows OS hosts: Windows.System.Powershell.PSReadline
    - Windows OS hosts: Windows.EventLogs.EvtxHunter(4104)
- Why use Powershell hunts?
    - PowerShell is frequently used for "living off the land" and masking activities because it is widely employed for administrative tasks.
    - Analyzing PowerShell is crucial for several reasons, including understanding common practices on the network, identifying potential malware, detecting persistence mechanisms, and recognizing C2 activities. These are important pieces of evidence necessary for understanding enemy behavior.
    - There has not been an easy way to quickly parse through the information, other than searching for more common keywords.
- PowerShell filtering recommendations:
    - **Nop**: This is used in conjunction with a command. It attempts to run a command with no profile (-noprofile).
    - **Connect:** Looking for non standard connection or attempts. Includes such commands as Test-Netconnection, -ConnectionUri, Get-NetTCPConnection, Test-MAPIConnectivity etc. Analysis can be further

refined by eliminating known scripts, processes.
    - **-e:** Looking to identify encoded commands that are being sent. This is sometimes being done malisciously but as well happens naturally on most systems. Different variations include -en, -enc, -enco etc.
    - **Command:** This will include command_includes and command. Both can be used for malicious and non-malicious purposes.
    - **Write:** Verifying information that is being written either to screen or desk., Can be leveraged for gathering information or installing malicious programs.
    - **Searching for 'SecureString', 'password', 'pass':** Looking for these strings is a quick way to find clear-text passwords in the console history file. This file does not need any special permissions to read, and is a good vector for adversaries to gain extra credentials.
    - **Searching for '.ps1':** Sometimes people are lazy and hardcode creds directly in scripts. It is a good idea to locate and investigate any script that lives on an endpoint.
    - **Searching for base64 strings:** Another good way to find creds with low level obfuscation.
    - **Searching for IPs:** Generally, there are many instances of Telnet, which is a bad practice for admins to use, but it allows for the detection of any malicious connections.

Amcache

- Velociraptor hunt artifacts

    - Windows OS hosts: Windows.Detection.Amcache

- Why should we use Amcache?

    - Application Compatibility Cache is used to improve the compatibility of applications. It as well keeps details on applications path, launch count and execution timestamps. It is also a possible avenue for attackers to hide their activities.
    - A large list of hashes can be collected and analyzed through malware scanning services like VirusTotal.

- Amcache filtering recommendations:

    - **SHA1:**

        - Compare to LOLBins

        - Analyze through a Malware analysis tool like VirusTotal

        - Find SHA1's that have been seen few times

        - Higher priority should be paid to hashes that have not been seen much

        - Focus on comparing hashes against known good hashes to narrow down your analysis.

            - A comprehensive list of known good hashes is available at https://www.circl.lu/services/hashlookup/.

    - **EntryName:**

        - Compare programs seen vs programs that should be installed

        - Look for:

            - names that have unusual spellings of common names (dElL vs Dell)
            - Non executable items

    - **Publisher:** Look for:

        - Typos of common names

        - Unusual or no Publisher
    - **OriginalFileName:**
        - Can be compared to Entryname items, they should match minus a few cases.

Netstat

- Velociraptor hunt artifacts
    - Linux.Network.Netstat
    - Windows.Network.Netstat
- Why use it?
    - Netstat provides a list of active network connection, routing, listening interfaces and protocols used in the network.
    - It is a crucial tool for several reasons, including listing established connections, identifying listening IP addresses and ports, and displaying associated processes and protocols. They can lead to identifing suspicious or malicious behaviour (i.e. C2) from a network point of view.
- Netstat filtering recommendations:
    - **Name** - Used to filter based on which process is currently running.
    - **CommandLine** - Can leverage to filter specific file location and/or additional parameters being mentioned.
    - **Hash** - Used to compare the process hash with provided IoC or run through a Malware Analysis tool like Virustotal.
    - **Authenticode** - (Trusted and Untrusted) If the process used by the netstat connection is untrusted, it will be mentioned. Could be an interesting way to filter only Untrusted processes and analyze the returned list.
    - **Laddr.IP** - (Source IP) Useful to exclude internal traffic from queries.
    - **Raddr.IP** - (Destination IP) Useful field for netstat since External IP destination can reveal suspicious or malicious patterns. Can use Regex to exclude all internal IP which appears. Once filtered for all external IP occurences, a separate analysis tool to quickly analyze the list can be leveraged.
    - **Laddr.Port / Raddr.Port** - Filters by specific ports number and range through Regex. Might help to locate suspicious connection based on port(s) usage.
    - **Timestamp** - Indicate the time when the netstat connection was started. It is useful as part of the investigation to recreate a timeline of events.

RDP / Other Remote Software

- Velociraptor hunt artifacts
    - Windows.EventLogs.RDPAuth
    - Windows.EventLogs.RDPClientActivity
- Other Remote Software (e.g., TeamViewer, AnyDesk, etc.):
    - Review logs for any remote access software in use, if available.
- Why use RDP hunts?
    - To extract Event Logs related to Remote Desktop sessions, logon and logoff.
    - Can be used to investigate password brute forcing
    - Investigate if a private host reached a public IP via RDP
    - The URLs or IPs can be investigated for suspicious use with tools like virus total / urlscan / shodan.
- Remarks

  - Some investigations will require correlation between multiple Microsoft Event ID to determine what actually happened.

- RDPAuth filtering recommendations:

  - **Description**

    - Filtering for failed logons can identify potential brute force.

  - **EventTime**

    - Filtering by event time can help reduce the amount of data to investigate.

  - **EventID**

    - Knowing the event ID can assist faster analyze data.

  - **LogonType**

    - Filter out "null" results
    - Filtering by different LogonTypes can help narrow down the search. Ex: Type 4 being batch logons.

  - **Message**

    - This line is where the most insight can be gained from the log. When analyzed in Elastic, reading the messages helps to better understand the events that occurred.

  - **ClientId**

    - This line can be helpful to filter down to a single host.

- RDP Client Activity filtering recommendations:

  - DestinationHost
  - Filtering out private ip range can identify external connections.

DNSCache

- Velociraptor hunt artifacts

  - Collects DNS cache entries

- Why use DNScache?

  - Stores recently resolved domain names and their corresponding IP addresses.

  - The URLs or IPs can be investigated for suspicious use with tools like virus total.

  - Remarks

    - DNS cache entries have Time To Live (TTL) values that are removed after expiry. It is required to run a new hunt somewhat regularly to fetch the latest cache contents.

- DNScache filtering recommendations:

  - **Name**

    - Remove local ip range in the query
    - Customize the query to remove amazon/google/microsoft/cloudflare results
    - Analyze suspicious names through Malware analysis tools such as VirusTotal / urlscan (website or API)

    - **Record**
        - One way to use it is to filter records after filtering names.
        - Remove private ip from this list
        - Analyze suspicious names through Malware analysis tools such as VirusTotal / urlscan (website or API)

Pslist

- Why use it
    - Pslist provides a snapshot of currently running processes on the host, displaying details such as process ID, parent process ID, memory usage, CPU usage, and more.
    - Pslist can highlight suspicious or malicious running programs, services or child processes. Unusual naming, unique occurrences or high resource usage can be indicators.
    - Each running processes in Pslist will have an unique hash associated, which can be analyzed with tools such as VirusTotal or Mandiant.
- Pslist filtering recommendations:
    - **Name** - Use to filter based on which process is currently running.
    - **CommandLine** - Filtering by specific file locations and/or additional parameters can be leveraged.
    - **Hash** - The process hash could be compared with the provided IOC or analyzed using a malware analysis tool like VirusTotal.
    - **Authenticode** - (Trusted and Untrusted) If the process used by the netstat connection is untrusted, it will be mentioned. Could be an interesting way to filter only Untrusted processes and analyze the returned list.

Browser History / Evidence of Download

- Velociraptor Hunt Artifacts
    - Linux.Collection.BrowserExtensions
    - Linux.Collection.BrowserHistory
    - Windows.Applications.Chrome.Extensions
    - Windows.Applications.Chrome.History
    - Windows.Applications.Edge.History
    - Windows.Applications.Firefox.History
    - Windows.Analysis.EvidenceOfDownload
    - Generic.Forensic.SQLiteHunter
- Why use Browser History?
    - A common vector for Initial access.
    - Often exploited through typo squatting, malicious scripts and enticing the download of dubious programs.
- Remarks
    - Useful when paired with IoC’s
    - Automation will greatly increase productivity
- Filtering Recommendations:
    - **URL**
        - Typos of common websites
        - Can eliminate internal websites
        - Websites that are unique to only a few hosts

  - **File Hash**
    - Files downloaded from unknown websites
    - Hashes that are not seen in services such as virus total
    - Files with macros attached or enabled (pptx,docx,xlsx)

Linux Parsing Artifacts

- Artifacts;
  - Custom.Linux.System.Autoruns (community artifact)
  - Custom.Linux.Sys.ParseSyslog
  - Custom.Linux.Sys.ParseWtmp
  - Custom.Linux.Sys.Dmesg
  - Custom.Linux.Sys.IPRoute
  - Custom.Linux.Sys.BootCmdLine
- When / Why to use them?
  - Once a target client has been narrowed down, it is fairly easy to pull the files mentioned above for manual grepping and analysis. These artifacts are designed to automate the process across large subnets.
  - Wtmp will show historic logins, including over the network.
  - Dmesg will show kernel message logs.
  - Syslog is a log protocol for log aggregation and depending on the .conf file will collect and aggregate logs from multiple sources.
  - IPRoute will show and parse the linux routing table, useful for network mapping and verification.
  - BootCmdLine can be useful to verify if certain settings are select that boot a machine into a potentially vulnerable configuration.
- Recommendations for use:
  - All of these artifacts are fairly simple and can be run with the default parameters.

Linux Process Environment variables

- When / Why to use it?
  - This will grab and parse all the '/proc/*/environ' enviroment variable files it can find. These files hold the environment variables for running processes on the system.
  - If the 'IncludeProcessInformation' flag is selected, it will join this information with information about the process that owns them through VQL's 'pslist()' plugin.
  - Most useful when a suspicious process has been identified, and requires further investigation without resorting to a full memory capture.
- Recommendations for use:

> ⚠ **WARNING** ⚠
> Do **NOT** run network wide.

- Identify a subset of hosts on a subnet, label them properly, and run this on them with relatively low resource parameters. It is not an intensive hunt, but because it is reading from '/proc' it requires special precautious.
- Confirm with leadership before running. Again, even though its not directly reading from process memory within '/proc', should still be handled with care.
- Ideally run with a PID in the glob; should also avoid running across all available processes in '/proc/'

Annex A VQL Examples

This annex provides a collection of simple VQL examples designed to assist with common threat hunting tasks. These queries serve as foundational templates for identifying potential IOCs and anomalous activities across various systems and environments. However, it is important to note that **these examples are not exhaustive** and should not be seen as a comprehensive approach to threat hunting. A robust, effective, and complete threat hunt requires a thorough understanding of the environment, context, and the specific objectives of the investigation, as well as continuous refinement of the queries and methodologies employed. These examples aim to help you get started but should be adapted and expanded based on the specific needs of your investigation and threat landscape.

Autoruns

Querying the Autoruns hunt results for cmd executables, .ps1, .vbs or .bat files, that are seen less than 5 times across the whole infrastructure.

```
let QUERY1 = SELECT `Image Path`, Entry, `Launch String`, count() AS Count
    FROM  hunt_results(artifact = 'Windows.Sysinternals.Autoruns', hunt_id = 'YOUR HUNT ID')
    WHERE `Launch String` =~ ".ps1"
    OR `Launch String` =~ ".vbs"
    OR `Launch String` =~ ".bat"
    OR `Launch String`=~ "cmd\.exe"
    GROUP BY `Launch String`

SELECT * FROM QUERY1
where Count <5
GROUP BY Entry
```

Other lists to look for in Autoruns are LOLBins. See URL for csv from lolbas.io for a list of LOLBins: hxxps://lolbas-project.github.io/api/lolbas.csv

RDP

RDP activity to external sources.

```
SELECT * FROM hunt_results(artifact = 'Windows.EventLogs.RDPClientActivity', hunt_id = 'YOUR HUNT ID')
WHERE DestinationHost =~ "^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$"
AND NOT cidr_contains(ip = DestinationHost, ranges = ["10.0.0.0/8", "127.0.0.0/8", "169.254.0.0/16", "192.168.0.0/16"])
GROUP BY DestinationHost
```

RDP connections from external sources.

```
SELECT * , count() as Count FROM hunt_results(artifact = 'Windows.EventLogs.RDPAuth', hunt_id = 'YOUR HUNT ID')
WHERE SourceIP =~ "^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$"
AND NOT cidr_contains(ip = SourceIP, ranges = ["10.0.0.0/8", "127.0.0.0/8", "169.254.0.0/16", "192.168.0.0/16"])
GROUP BY SourceIP
ORDER BY Count
```

Amcache

Looking for hashes that have been seen 10 or less times across the hunt. This can then be put through Mandiant/Virustotal

```
LET QUERY1 = SELECT *, count() as Count FROM hunt_results(artifact='Windows.Detection.Amcache',
hunt_id='YOUR HUNT ID')
    GROUP BY SHA1

SELECT * FROM QUERY1
    WHERE Count < 11
    ORDER BY Count
```

Amcache can also be used to look for LOLDrivers. Link to list of LOLDrivers below:
hxxps://www.loldrivers.io/api/drivers.csv

To look for vulnerable drivers:

1. Upload hashes to VirusTotal using the "VirusTotal-Pipeline" (See WikiJS for more info on the tool)
2. Pull down the results to local host
3. Extract the hashes from LOLDrivers csv file
4. Run ```grep -r -f $lolDriverHashes $VTResults.csv
5. If there are matches, start investigation on the driver.

PSReadline / ScriptBlock

Reads \AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt. Below is a query for some common keywords but is not all inclusive

```
SELECT Line, Username, Fqdn, count() as Count FROM hunt_results(artifact =
'Windows.System.Powershell.PSReadline', hunt_id = 'YOUR HUNT ID')
   WHERE Line =~ '(?i)(-e$|-nop|connect|command|write|IEX|IWR|Invoke|username|password|secret|public)'
   GROUP BY Line
```

Netstat Enriched CommandLine

Uses Netstat enriched to look for Unique CommandLines seen from the hunt and organises them based upon how many seen. Eliminates internal traffic as well as a lot of Microsoft traffic. Still needs further tuning.

```
Select CommandLine, `Raddr.IP`, count() as Count FROM hunt_results(artifact =
'Windows.Network.NetstatEnriched/Netstat', hunt_id = 'YOUR HUNT ID')
WHERE `Raddr.IP` =~ "[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}"
AND NOT cidr_contains(ip = `Raddr.IP`, ranges = ["10.0.0.0/8", "192.168.0.0/16"])
AND `Laddr.IP` != "0.0.0.0" AND `Laddr.IP` != "127.0.0.1"
AND Hash.SHA256 != "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855"

//Microsoft CIDRs
AND NOT cidr_contains(ip = `Raddr.IP`, ranges = ["40.126.128.0/17", "40.127.0.0/16", "52.148.0.0/14",
"52.145.0.0/16", "52.146.0.0/15", "52.152.0.0/13", "52.160.0.0/11", "20.0.0.0/11", "13.96.0.0/13", "13.64.0.0/11",
"13.104.0.0/14", "40.126.128.0/17", "40.127.0.0/16", "40.64.0.0/13", "20.34.0.0/15", "20.40.0.0/13", "20.64.0.0/10",
"20.48.0.0/12", "20.36.0.0/14", "20.33.0.0/16", "20.128.0.0/16"])
GROUP BY CommandLine
ORDER BY Count
```

Lists of external IP's seen 10 or less times on the network. Possibly usable from within Shiva/Toolage API's.

```
LET QUERY1 = Select CommandLine, `Raddr.IP`, count() as Count FROM hunt_results(artifact =
'Windows.Network.NetstatEnriched/Netstat', hunt_id = 'YOUR HUNT ID')
 WHERE `Raddr.IP` =~ "[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}"
 AND NOT cidr_contains(ip=`Raddr.IP`, ranges=["10.0.0.0/8", "192.168.0.0/16"])
 AND `Laddr.IP` != "0.0.0.0" AND `Laddr.IP` != "127.0.0.1"
 AND Hash.SHA256 != "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855"

 //Microsoft CIDRs
 AND NOT cidr_contains(ip=`Raddr.IP`, ranges=["40.126.128.0/17", "40.127.0.0/16", "52.148.0.0/14",
"52.145.0.0/16", "52.146.0.0/15", "52.152.0.0/13", "52.160.0.0/11", "20.0.0.0/11", "13.96.0.0/13", "13.64.0.0/11",
"13.104.0.0/14", "40.126.128.0/17", "40.127.0.0/16", "40.64.0.0/13", "20.34.0.0/15", "20.40.0.0/13", "20.64.0.0/10",
"20.48.0.0/12", "20.36.0.0/14", "20.33.0.0/16", "20.128.0.0/16"])
 GROUP BY `Raddr.IP`

SELECT * FROM QUERY1
WHERE Count < 10
```

Linux Bash History

View what devices the linux machine is connecting to. Used as a baseline for possible maliscious connectivity between internal devices to external sources. Additional information that could be ignored 10.0.0.0/8, internal domains, known connection points.

```
SELECT * FROM hunt_results(artifact = 'Linux.Sys.BashHistory', hunt_id = 'YOUR HUNT ID')
WHERE Line =~ "^connect|^telnet|^ssh|sudo connect|sudo telnet|sudo ssh"
ORDER BY Fqdn
```

Possible ways of browsing websites in bash

```
SELECT * FROM hunt_results(artifact='Linux.Sys.BashHistory', hunt_id='YOUR HUNT ID')
WHERE Line =~ "^wget|^curl|^nc|^netcat"
ORDER BY Fqdn
```

PSList

Pull a limited anmount of running hashes that are seen 10 or fewer times across the network. This allows a limited amount to be used with VirusTotal

```
LET QUERY1 = SELECT Hash.MD5, Exe, count() as Count, ClientId
FROM hunt_results(artifact = 'Windows.System.Pslist', hunt_id = 'YOUR HUNT ID')
   GROUP BY Hash.MD5

SELECT * FROM QUERY1
WHERE Count <= 10
```

Coverage assessment using GetADComputers

GetADComputers does exactly as it says. This VQL snippet should be ran to identify systems without agents WITHIN an active directory domain.

- Note - This will not identify any missing non-domain joined machines, leverage Local admins for resolving any conflicts not domain joined.

```
//Gets all computers info from GetADComputerArtifact where enabled
LET ADCOMPUTERS <= SELECT Name, SamAccountName, OperatingSystem, LastLogonDate
FROM hunt_results(
  artifact='playbook.Custom.ActiveDirectory.GetADComputers',
  hunt_id='H.ID')
Where Enabled = true

//Gets hostname from clients() and turns it into an array to compare
Let InstalledClients <= Select os_info.hostname as Name from clients()
LET InstalledClients <= array(_=InstalledClients).Name

//Compares list of enabled computers showing only those not in clients()
SELECT * From ADCOMPUTERS
Where NOT Name in InstalledClients
```

Coverage Assessment using ARP Caches *Depreciated*

The ARP cache contains local IP addresses known to each hosts, which can be compared to the IPs of Velociraptor agents to itendify the IPs of devices without agents. These IPs will naturally include netwrok devices, printers, etc which should not have agents, but they still provide an indication of coverage as the number of such devices should be relatively small compared to workstations and servers.

Step one is selecting the unique IP addresses from the ARP chaches of hosts on the network. Note that different hunts are required for different OS. Step two is collecting the IP addresses associated with each velo agent, which are available from the clients() plugin. Because we have IP address lists from multiple hunts, they must be combined into a single array in step 3. Finally, the list of ARP IPs is filtered to remove the IPs of velociraptor agents in step 4.

Its important to note this is NOT definitive and there will be overlap due to DHCP pools, results may vary.

```
LET WinArpData <= SELECT RemoteAddress as IP
                FROM hunt_results(
                    artifact='Windows.Network.ArpCache',
                    hunt_id='H.ID')
                    WHERE not State=~"Incomplete|Unreachable"

LET LinxArpData <= SELECT IP_address as IP
                    FROM hunt_results(
                        artifact='Linux.Proc.Arp',
                        hunt_id='H.ID')
                        WHERE Flags=~"0x2"

-- Combine all ARP data (IP and MAC pairs)
LET AllArpData <= SELECT * FROM chain(
    a={SELECT IP FROM WinArpData},
    b={SELECT IP FROM LinxArpData}
)
GROUP BY IP

-- Select IPs in arp caches which are not associated with a velo agent
LET IPregex <= "\\b(?:\\d{1,3}\.){3}1\\b|\\b(?:\\d{1,3}\.){3}255\\b" --filter gateways which end in '.1' and brodcast
addresses ending in '.255'
LET CLIENTIPs <= SELECT split(sep_string=":", string=last_ip)[0] as IP from clients() --parses last IP field then
appends it to an array to search
LET CLIENTIPs <= array(_=CLIENTIPs).IP

--Filter ips and show data
SELECT * FROM AllArpData
WHERE not IP =~ IPregex
AND NOT IP in CLIENTIPs
```

Browser History - SQLiteHunter

Globs Regex by OS

- Windows - C:\Users\*\AppData***history*

- Linux - /home/*/{.mozilla,.config}/**/{places,history,manifest}

This should capture sites visited, downloads using the browser, and extensions added. Based on big amounts of data, it is recommended to download the hunt results and parse the data using other tools such as 'LibreOffice' or grep. Note: Ensure you deselect all tabs and only select the "Browser" Tab. The "SQLITE_ALWAYS_MAKE_TEMPFILE" tab should also stay at default.

Examples for grep:

- `grep -ow -f './myWordList.txt' 'AllChromeBrowserHistory.csv'
    - Run wordlist against csv results
- cut -d ',' -f<your field number here> 'AllChromeBrowserHistory.csv' | sort | uniq > filteredURLs.txt
    - Return only the URLs from the csv to make the file smaller and easier to work with.

Anomaly Detection with Aggregation

VQL provides a number of aggregation functions to apply to grouped items, including:

- sum()
- count()
- min()
- max()

Here is an example of sum() and count() to find programs which have been executed on a few hosts or few times. We expect malware to only exist on small number of hosts in all but the most extreme cases, with the number of executions smaller than legitimate programs. This query is also useful for determining normal patterns of usage on the network:

```
SELECT Executable, sum(item=RunCount) as RunTotal, count() as HostCount
FROM hunt_results(
    artifact='Windows.Analysis.EvidenceOfExecution/Prefetch',
    hunt_id='YOUR HUNT ID')
GROUP BY Executable
ORDER BY RunTotal
```

The min() and max() functions are particularly usful for timstamps, for example finding the first or last time a program was run.

```
SELECT min(item=LastRunTimes[-1]) as FirstPrefetchEntry, Fqdn
FROM hunt_results(

    artifact='Windows.Analysis.EvidenceOfExecution/Prefetch',
    hunt_id='YOUR HUNT ID')
GROUP BY Fqdn
ORDER BY FirstPrefetchEntry
```

This example uses the min() function to find the oldest prefetch entry on each host.

Windows Defender Exclusions

Defender can often be found in PS Readlines if the history is intact and the changes weren't made with the GUI. It more reliable to find these changes in the registry, but the location varies. Seaching the registry with the glob `HKEY_LOCAL_MACHINE\**\Exclusions\**` in a registry parsing artifact.

Annex B Questionnaire Example

Recce Questions

Organization Information

- Location(s) of endpoints/servers.
- Operations Hours (Do they shut computers at end of day?)
- Identified Accounts: Sys Admins, Trusted Personnel, System Owners.
    - How is privileged access controlled and monitored?
- What are the purposes for these networks?
- What types of sensitive or critical data are stored or processed on these networks?
- IT security policy description (what are their practical implementations?)
- Third party dependencies? (IT service Provider, Connected Networks)
    - What monitoring and access controls are in place for these vendors?
- ISP providers? Link speeds? Bandwidth Constraints?
- Previous/Upcoming pen test & vulnerability assessments? (Reports)

Request/ Gather Documentation & Network Maps

- Asset List (Hardware and Software):
- List of servers and endpoints (Server & support systems inventory list)
- Estimated number of switching, routing, firewall, and encryption devices.
- Host Baseline Image, what OS do you use?
- List and visual representations of: Netblocks (internal/external), assigned IP addresses, domain names, network zones, vlans, firewalls, Proxies. (Network Maps)
- What are your critical assets/infrastructure, services? Backups?
- User systems management architecture description and involved components. (Active Directory)
- How the central management and software deployment is done on user infrastructure?
- Are there remote (Internal/External) access capabilities (i.e. VPN, TeamViewer, RDP, SSH, VNC) and what authentication method(s) is/are used?
- Centralized Mobile Device management?
- What cloud providers are used in your organization? (e.g. Azure, AWS, Office365). Cloud based file sharing services (OneDrive, DropBox, Google Drive etc.)
- Do you have any internet accessible points of presence? (Public facing websites)
- What is their purpose?

Vulnerability Management

- Anti-Virus solutions/configs, how their telemetry is collected and processed?
- Any Cyber Defence Measures in place?
- Are there specific GPOs for security hardening? How frequently are GPOs reviewed and updated?
- Are there any Software Restriction Policies (SRPs) implemented to control which applications can run on the network?
- List of available monitoring/logging capability, SIEM solutions.
- Can logs be injected into our SIEM?

Annex C Infrastructure Baseline

About

This doc is inteded to give an overview of the network as we see it **during** the hunt. It can be enriched with data given from the network admins but should be mainly populated using data gathered with Velociraptor. Data here represents the **intended** tooling and should be confirmed with administrators, if needed.

Anyone who has access to this doc can edit it.

This includes but is not limited to:

- VPNs
- Remote tooling
- Local AVs
- Other local security services (EDR, XDR, Zabbix, etc.)
- SIEM / EDR dashboards / other monitoring
- Public facing services
- Other

Below is an example of how a Baseline might look once the initial Investigation has been completed. This document would be used as a reference to look back upon when further investigating the network. This would help keep track of common applications that can be found, ports used, IPs seen in network activity, and overall save time on investigating things unnecessarily.

Certain tools and scripts can be used to speed up the Baselining such as the python script [host_diff.py](/Other Tools/hosts_diff.py). Which takes a list given from the System administrators of the network you are investigating and allows you to compare this list to the total number of devices you can monitor on the network using Velociraptor. This will help to establish what level of coverage you have on the network, as well as help to find if there are issues with any of the Velociraptor deployments on the network.

Example Network Baseline

Replace title / header with your network name.

## Number of Devices

Total (116)
Windows 10 pro (46)
Windows 11 pro (70)

## Operating Systems

Windows

## Domains

example.lv

## Networks

192.168.0.0/24 (27)
192.168.7.0/24 (2)

192.168.9.0/24 (4)
192.168.110.0/24 (46)
192.168.210.0/24 (19)
192.168.215.0/24 (64)
192.168.216.0/24 (1)
192.168.217.0/24 (13)
192.168.218.0/24 (11)
192.168.230.0/24 (1)

## Antivirus

Base Publisher: ETM professional control

## EDR / XDR

ESET Endpoint Security
ESET Management Agent
Cortex XDR 8.4.0 Palo alto

## VPN

Global protect Palo alto (13)
u-link VPN Client (1)
Weidmüller u-link VPN Client (1)
ads-tec VPN components (1)

## Remoting Tools

Teamviewer TightVNC

## Office Suites

Outlook Onedrive Gmail
Google slides
Google sheets
Google docs
Microsoft Office
Microsoft Teams
Zoom
Adobe Acrobat
signal
skype
7zip
peazip
webex cisco (conference)
vlc media player
sentinel runtime (license manager)
QGIS (GIS)
gotomeeting
whatsapp
clickshare
WinSCP
Telegram
keepass
AWP IDEMIA smart card/biometrics digital security

SAP Business explorer (business intelligence suite for planning report analysis)
Latvian (Aprostrofs v1.9; punkts) latvian characters and punctuation software
eparaksts token signing (digital signnature)
medus App (electronic signature services)
Latvia eID middleware (connecting sofware for electronic id system apps that require digital auth)
Realtek dash client (software component related to realtek network interfaces)
Zulu 8.0 (cartography app GIS)
awp (smarcard and digital sec)

## Browsers

Google chrome
Firefox
Questionmark Secure Browser

## Administration Tools

Snow Inventory (4)

## Monitoring Tools

Wireshark 4.0.10 (1)
Nmap 7.95 (1)
Npcap (1)

# Servers

No Servers found

# Connections

## Network Listeners (116 agents)

Name	IP	Listening on	Port	Application
UserWS1.example.lv	192.168.215.86	0.0.0.0	49671	ekrn.exe
UserWS2.example.lv	192.168.215.75	0.0.0.0, ::	49690	services.exe
Server10.example.lv	192.168.0.244	0.0.0.0, ::	161	snmp.exe
Server97.example.lv	192.168.217.79	192.168.217.79	49690	

## Benign Connections (116 agents)

Excluding internal IP connections and many Microsoft Apps to "81.198.85.134", a SIA Tet address.

Process Name	Remote IP	Remote Port	Service Description
ekrn.exe	91.228.165.159	443, 8883	ESET antivirus software
ERAAgent.exe	91.228.165.148	443, 8883	ESET Remote Administrator Agent
ms-teams.exe	81.198.85.134	443	Microsoft Teams
msedgewebview2.exe	81.198.85.134	443	Embedding web content in desktop applications
Spotify.exe	81.198.85.134	443, 4070	Spotify streaming service

Process Name	Remote IP	Remote Port	Service Description
cyserver.exe	81.198.85.134	443	Palo Alto Networks' Traps Reporting Service
Skype.exe	81.198.85.134	443	Skype communication
chrome.exe	81.198.85.134	5228	Push notifications from the browser
Microsoft.SharePoint.exe	52.111.243.110	443	Microsoft SharePoint
OneDrive.exe	20.199.120.85	443	Microsoft OneDrive cloud storage
ReadyManager.ServiceHost.exe	20.101.249.34	9443	PC software for handling data from heat, cooling, and water devices
TeamViewer_Service.exe	188.172.198.144	5938	TeamViewer remote access
Signal.exe	13.248.212.111	443	Signal messaging app

## Anamolies

Many users have TiemViewer.exe installed with active connections at the time of the hunt, all connected to the expected port of 5938, except for one user, who is connected on remort port 3128.

Name	Pid	Laddr.IP	Laddr.Port	Raddr.IP	Raddr.Port	Fqdn	Statu
TeamViewer_Service.exe	7272	192.168.210.141	59763	81.198.85.148	3128	user1.example.lv	ESTAI

These hosts did not align with the provided list from the administrators, but were found in the network.

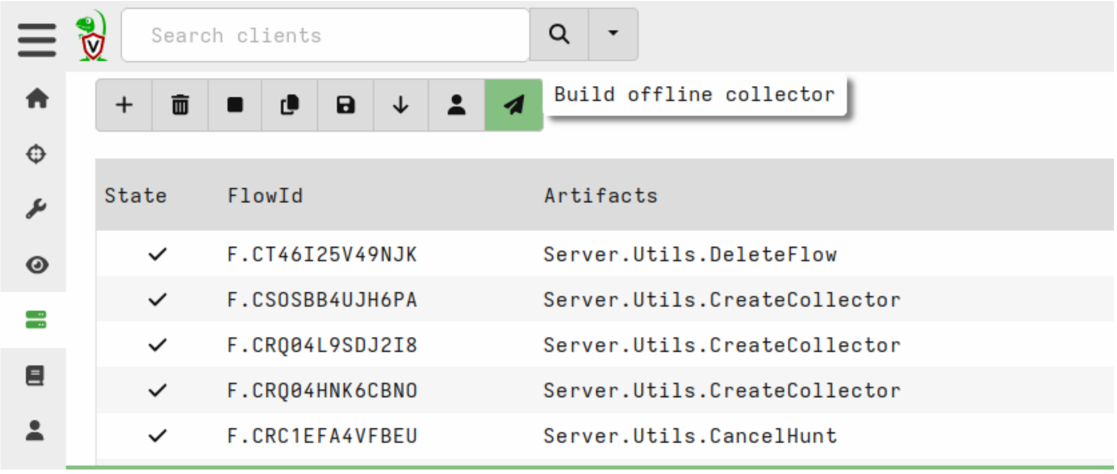
FQDN	IP Address	OS	Last Seen
userX.example.lv	192.168.215.232	Windows 10	2024-09-18T06:07:31.529539Z
userX.example.lv	192.168.230.76	Windows 11	2024-09-18T06:07:31.526457Z
userX.example.lv	192.168.0.70	Windows 10	2024-09-18T06:07:31.523036Z
userX.example.lv	192.168.0.79	Windows 11	2024-09-13T11:57:39.70845Z
userX.example.lv	192.168.0.150	Windows 10	2024-07-31T10:43:46.435824Z
userX.example.lv	192.168.0.136	Windows 11	2024-07-25T11:20:18.392691Z

Annex D Creating and Using an Offline Collector

The Velociraptor offline collector is designed for scenarios where it is not possible to deploy an active agent on the device under investigation but it is possible to execute a binary. For example, in air-gapped networks or situations where responsible parties do not support agent installation. With the offline collector, it is possible to gather the same actions as with an active agent. The process of collecting data is as follows:

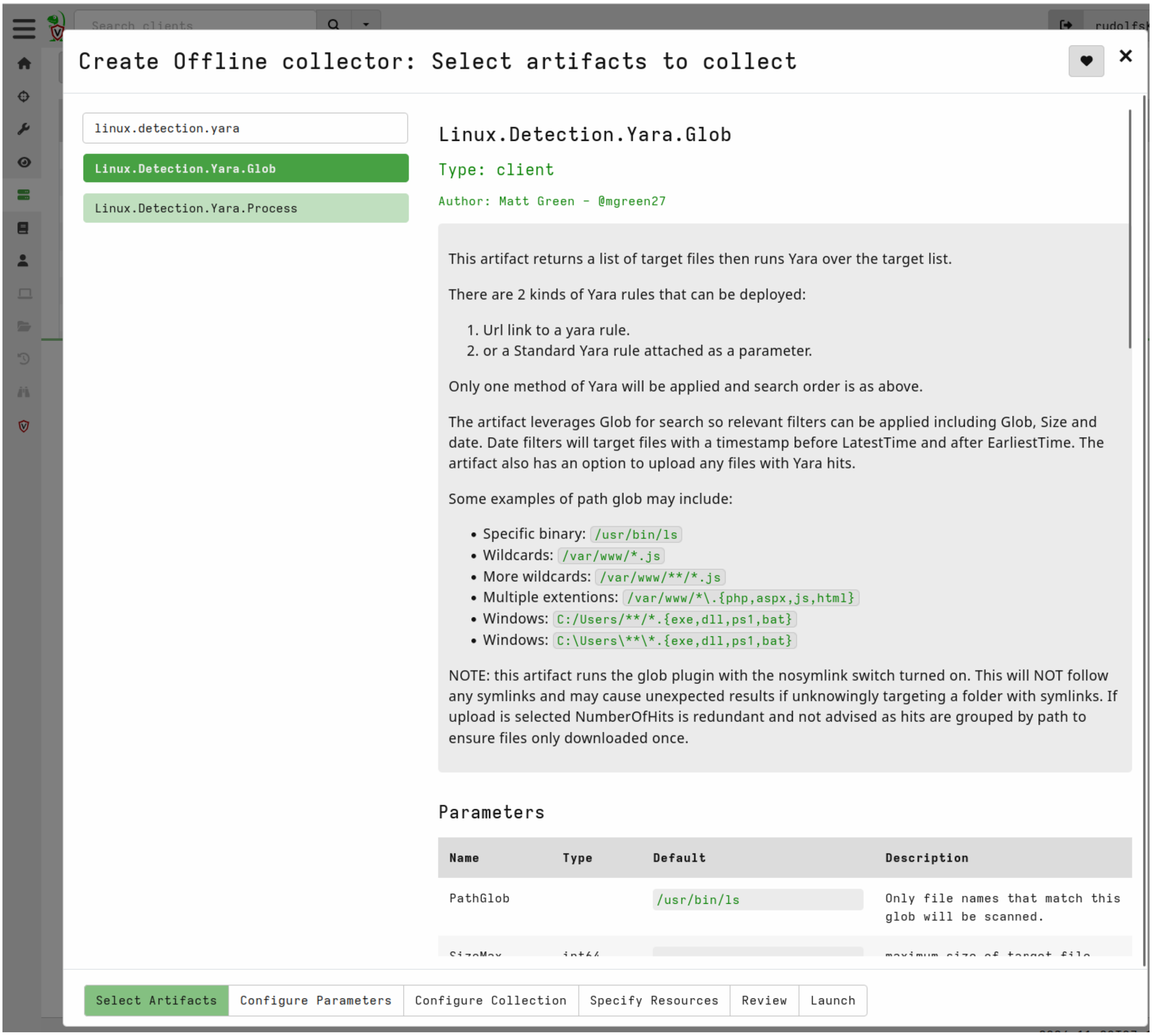
Creating the Offline Collector

- Log in to any Velociraptor server and navigate to "Server Artifacts."
- Click "Build offline collector."



State	FlowId	Artifacts
✓	F.CT46I25V49NJK	Server.Utils.DeleteFlow
✓	F.CSOSBB4UJH6PA	Server.Utils.CreateCollector
✓	F.CRQ04L9SDJ2I8	Server.Utils.CreateCollector
✓	F.CRQ04HNK6CBNO	Server.Utils.CreateCollector
✓	F.CRC1EFA4VFBEU	Server.Utils.CancelHunt

- Select the necessary "Artifacts" that will be executed on the target system. The choice depends on the target device (Windows / Linux) as well as the incident. For example, in the case of a web server investigation, it would be necessary to use Linux.Detection.Yara.Glob with web-shell rules. There is **no** limit to how many artifacts can be added to a single offline collector.



Create Offline collector: Select artifacts to collect

linux.detection.yara

- Linux.Detection.Yara.Glob
- Linux.Detection.Yara.Process

Linux.Detection.Yara.Glob

Type: client

Author: Matt Green - @mgreen27

This artifact returns a list of target files then runs Yara over the target list.

There are 2 kinds of Yara rules that can be deployed:

1. Url link to a yara rule.
2. or a Standard Yara rule attached as a parameter.

Only one method of Yara will be applied and search order is as above.

The artifact leverages Glob for search so relevant filters can be applied including Glob, Size and date. Date filters will target files with a timestamp before LatestTime and after EarliestTime. The artifact also has an option to upload any files with Yara hits.

Some examples of path glob may include:

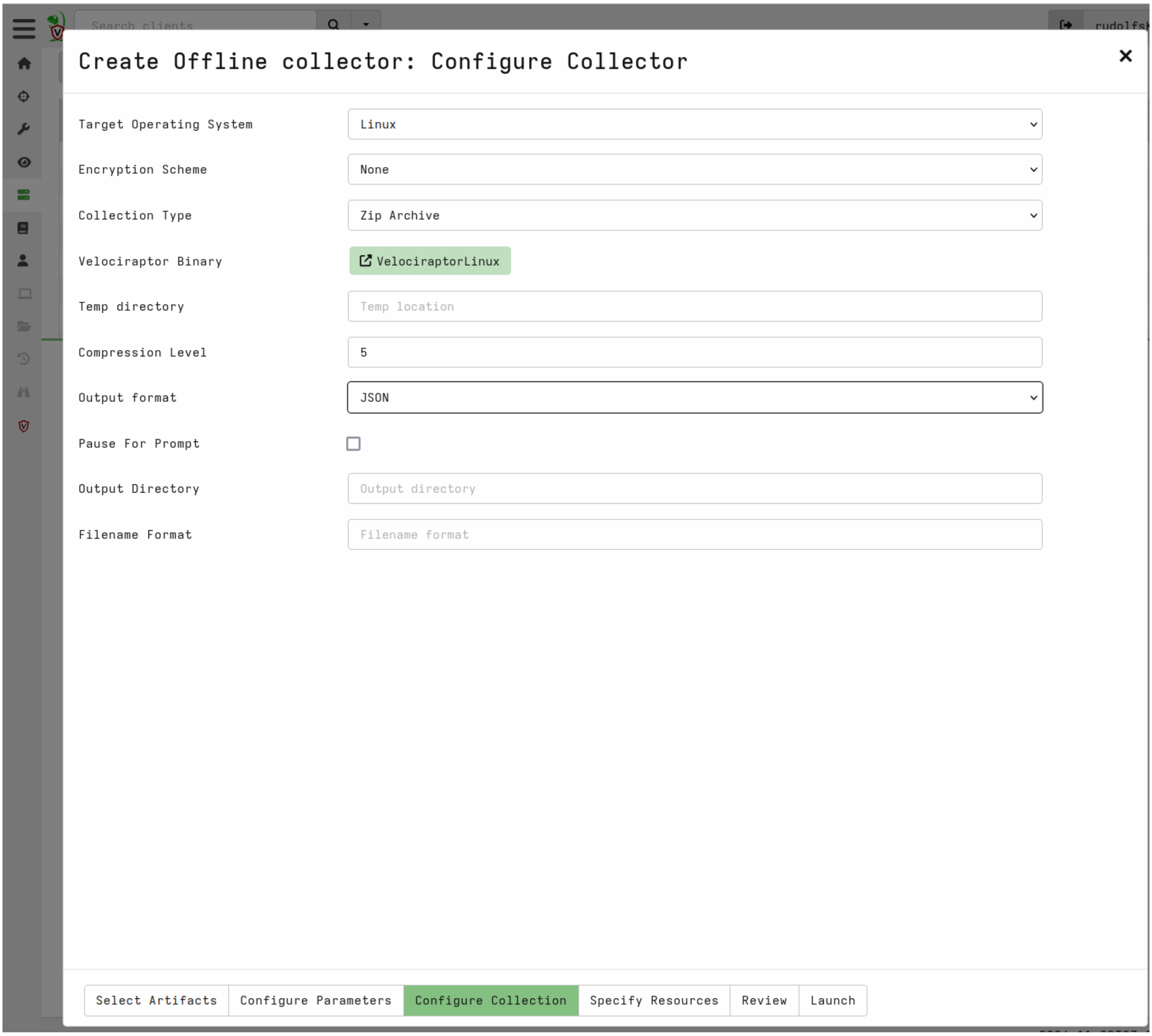
- Specific binary: `/usr/bin/ls`
- Wildcards: `/var/www/*.js`
- More wildcards: `/var/www/**/*.js`
- Multiple extentions: `/var/www/*\.{php,aspx,js,html}`
- Windows: `C:/Users/**/*.{exe,dll,ps1,bat}`
- Windows: `C:\Users\**\*.{exe,dll,ps1,bat}`

NOTE: this artifact runs the glob plugin with the nosymlink switch turned on. This will NOT follow any symlinks and may cause unexpected results if unknowingly targeting a folder with symlinks. If upload is selected NumberOfHits is redundant and not advised as hits are grouped by path to ensure files only downloaded once.

Parameters

Name	Type	Default	Description
PathGlob		/usr/bin/ls	Only file names that match this glob will be scanned.
SizeMax	int64		maximum size of target file

- If needed, the selected artifacts can be configured in the "Configure Parameters" section. For example, Yara artifacts can be supplemented with Yara rules.
- In the "Configure Collection" section, collector settings can be specified, such as target OS, result file type (zip, SFTP upload, etc.), password encryption, naming, and more. By default, the collector will save results in a .zip file named Collection-%FQDN%-%TIMESTAMP%.



Create Offline collector: Configure Collector

Target Operating System: Linux

Encryption Scheme: None

Collection Type: Zip Archive

Velociraptor Binary: VelociraptorLinux

Temp directory: Temp location

Compression Level: 5

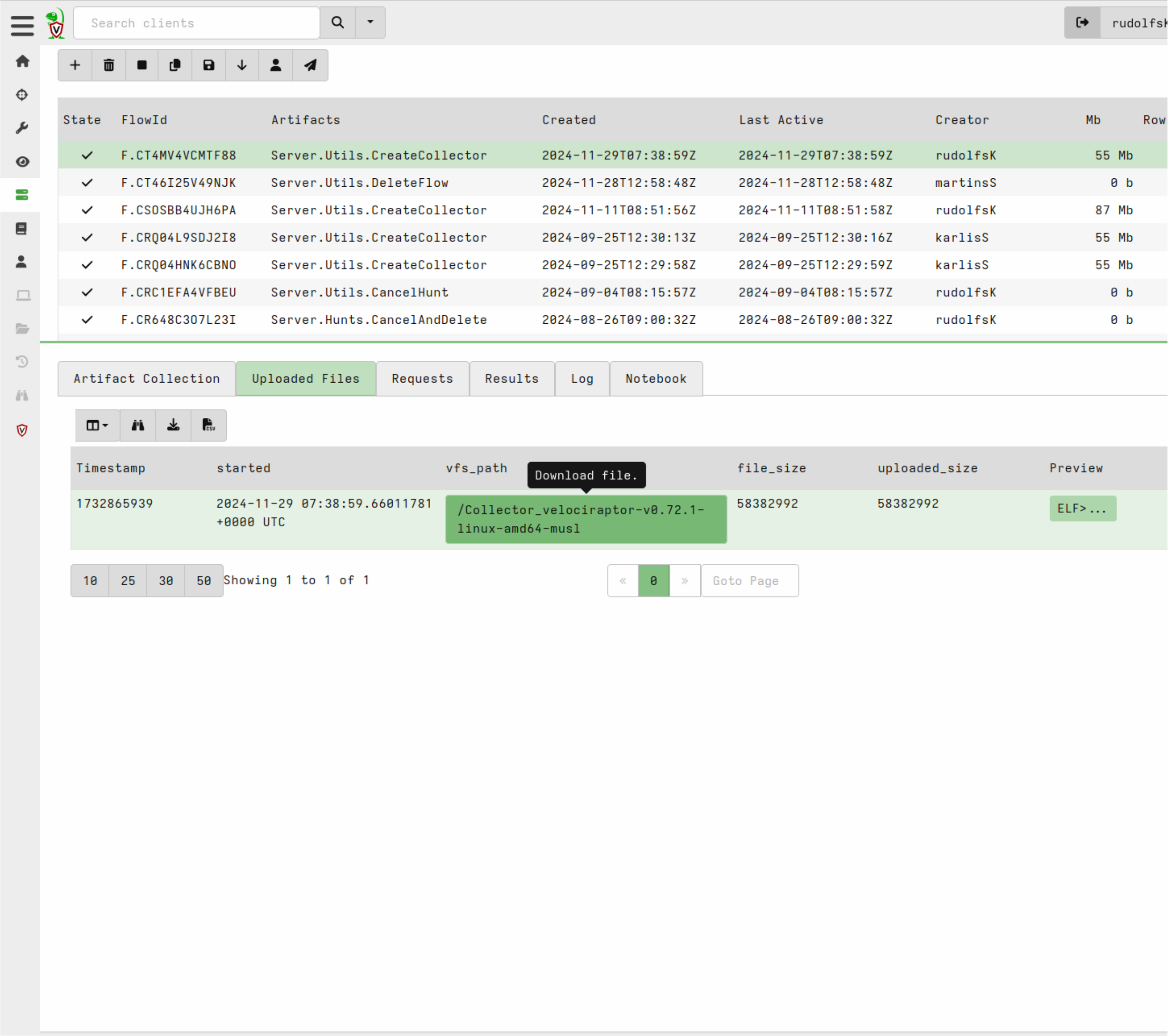
Output format: JSON

Pause For Prompt: □

Output Directory: Output directory

Filename Format: Filename format

- In the "Specify Resources" section, it is possible to set a CPU threshold above which the Velociraptor binary will not execute. For instance, setting it to 70% means the binary **will not execute** if the CPU load exceeds 70%.
- Once everything is configured, click "Launch."
- The result will be a binary that can be downloaded from the "Uploaded Files" section.



State	FlowId	Artifacts	Created	Last Active	Creator	Mb	Row
✓	F.CT4MV4VCMTF88	Server.Utils.CreateCollector	2024-11-29T07:38:59Z	2024-11-29T07:38:59Z	rudolfsK	55 Mb	
✓	F.CT46I25V49NJK	Server.Utils.DeleteFlow	2024-11-28T12:58:48Z	2024-11-28T12:58:48Z	martinsS	0 b	
✓	F.CSOSBB4UJH6PA	Server.Utils.CreateCollector	2024-11-11T08:51:56Z	2024-11-11T08:51:58Z	rudolfsK	87 Mb	
✓	F.CRQ04L9SDJ2I8	Server.Utils.CreateCollector	2024-09-25T12:30:13Z	2024-09-25T12:30:16Z	karlisS	55 Mb	
✓	F.CRQ04HNK6CBNO	Server.Utils.CreateCollector	2024-09-25T12:29:58Z	2024-09-25T12:29:59Z	karlisS	55 Mb	
✓	F.CRC1EFA4VFBEU	Server.Utils.CancelHunt	2024-09-04T08:15:57Z	2024-09-04T08:15:57Z	rudolfsK	0 b	
✓	F.CR648C307L23I	Server.Hunts.CancelAndDelete	2024-08-26T09:00:32Z	2024-08-26T09:00:32Z	rudolfsK	0 b	

Timestamp	started	vfs_path	file_size	uploaded_size	Preview
1732865939	2024-11-29 07:38:59.66011781 +0000 UTC	/Collector_velociraptor-v0.72.1-linux-amd64-musl	58382992	58382992	ELF>...

10 25 30 50 Showing 1 to 1 of 1

« 0 »

Transferring the Collector to the Target Device

This can be achieved in various ways, such as:

- Using a cloud storage service such as Google Drive, OneDrive, or Dropbox to upload the file and share the link with the administrators.
- Sharing files directly via email if the file size is manageable and does not exceed email attachment limits.
- Hosting the file on an internal web server or secure FTP server and providing access details to authorized users.
- Using a USB drive or other physical media for offline sharing in secure environments.

Running the Collector on the Target Device

- On Linux devices: `sudo /path/to/velociraptor-binary`
- On Windows: right-click the binary and select "Launch as Administrator."
- The result will be a `.zip` archive, which the responsible administrator must send back to the Threat Hunt team.

Importing Data into the Velociraptor Server

- The `.zip` archive created in the previous step needs to be uploaded to your Velociraptor server:

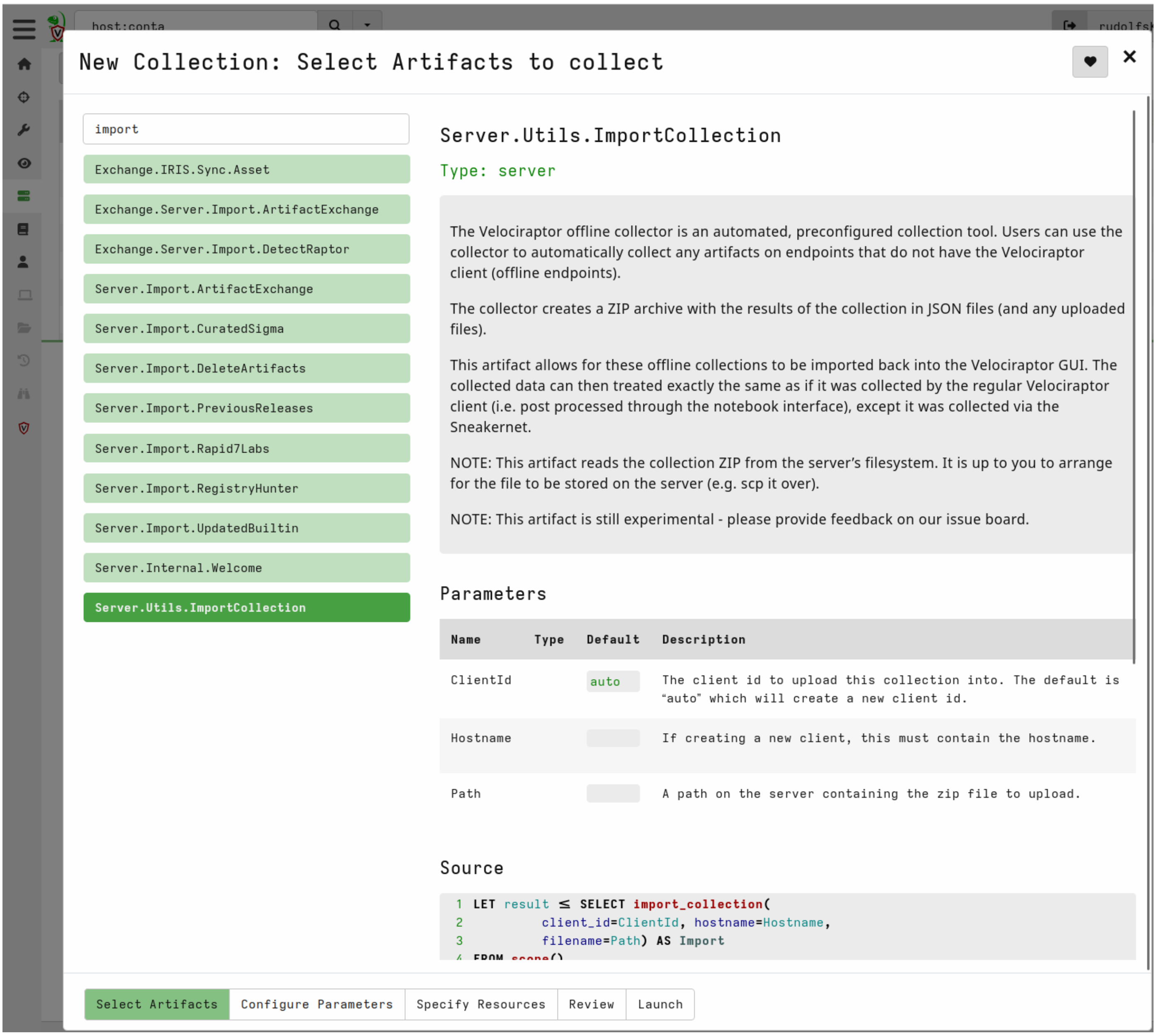
- Upload the zip file to the /tmp folder:

```
scp ./Collection-FQDN-TIMESTAMP.zip irvelo:/tmp/
```

- Copy the .zip file to the Velociraptor folder:

```
ssh irvelo sudo cp /tmp/Collection-FQDN-TIMESTAMP.zip /home/velo/velociraptor-docker/velociraptor
```

- Navigate to "Server Artifacts," click "New Collection," and select the "Import Collection" artifact.



New Collection: Select Artifacts to collect

import

- Exchange.IRIS.Sync.Asset
- Exchange.Server.Import.ArtifactExchange
- Exchange.Server.Import.DetectRaptor
- Server.Import.ArtifactExchange
- Server.Import.CuratedSigma
- Server.Import.DeleteArtifacts
- Server.Import.PreviousReleases
- Server.Import.Rapid7Labs
- Server.Import.RegistryHunter
- Server.Import.UpdatedBuiltin
- Server.Internal.Welcome
- Server.Utils.ImportCollection

Server.Utils.ImportCollection

Type: server

The Velociraptor offline collector is an automated, preconfigured collection tool. Users can use the collector to automatically collect any artifacts on endpoints that do not have the Velociraptor client (offline endpoints).

The collector creates a ZIP archive with the results of the collection in JSON files (and any uploaded files).

This artifact allows for these offline collections to be imported back into the Velociraptor GUI. The collected data can then treated exactly the same as if it was collected by the regular Velociraptor client (i.e. post processed through the notebook interface), except it was collected via the Sneakernet.

NOTE: This artifact reads the collection ZIP from the server’s filesystem. It is up to you to arrange for the file to be stored on the server (e.g. scp it over).

NOTE: This artifact is still experimental - please provide feedback on our issue board.

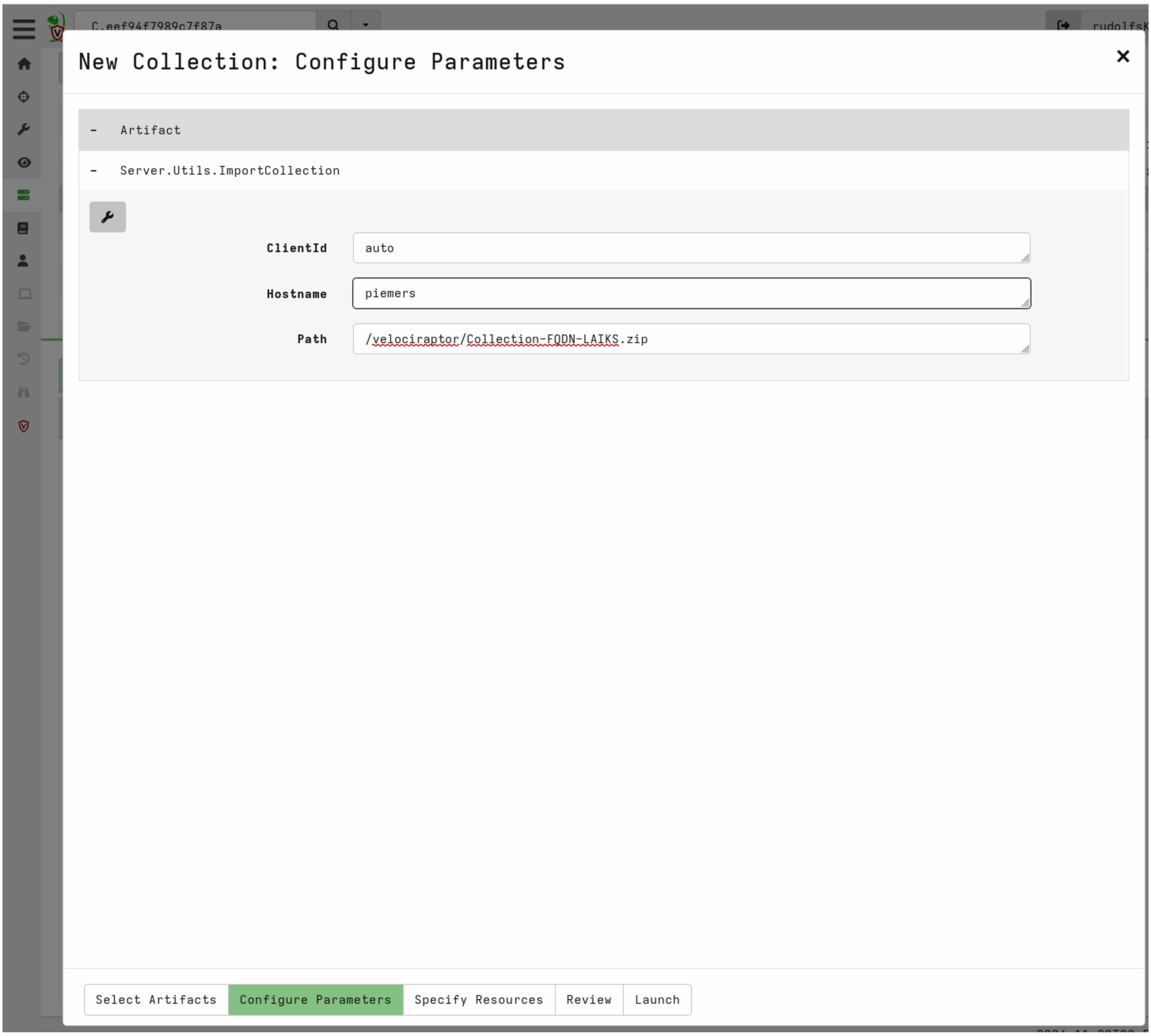
Parameters

Name	Type	Default	Description
ClientId		auto	The client id to upload this collection into. The default is “auto” which will create a new client id.
Hostname			If creating a new client, this must contain the hostname.
Path			A path on the server containing the zip file to upload.

Source

```
LET result ≤ SELECT import_collection(
          client_id=ClientId, hostname=Hostname,
          filename=Path) AS Import
FROM scope()
```

- In the "Configure Parameters" section, specify the location of the .zip file with /velociraptor before it.



New Collection: Configure Parameters

- Artifact

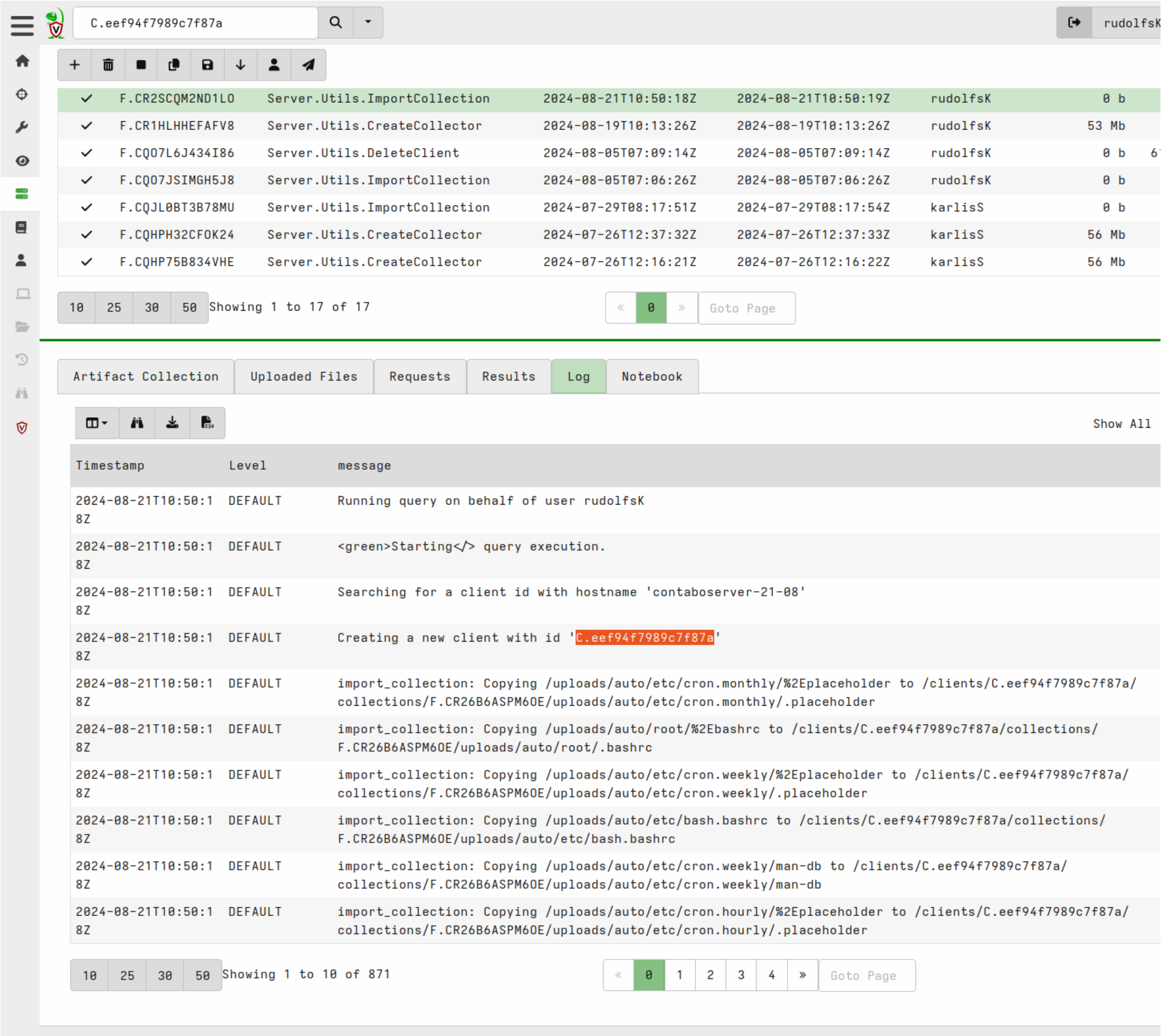
- Server.Utils.ImportCollection

ClientId: auto

Hostname: piemers

Path: /velociraptor/Collection-FQDN-LAIKS.zip

- Click "Launch"! By reviewing the "Log" section, it will be possible to see the new client's identifier. Entering it into the search will allow you to find the collected data.



✓	F.CR2SCQM2ND1LO	Server.Utils.ImportCollection	2024-08-21T10:50:18Z	2024-08-21T10:50:19Z	rudolfsK	0 b	
✓	F.CR1HLHHEFAFV8	Server.Utils.CreateCollector	2024-08-19T10:13:26Z	2024-08-19T10:13:26Z	rudolfsK	53 Mb	
✓	F.CQ07L6J434I86	Server.Utils.DeleteClient	2024-08-05T07:09:14Z	2024-08-05T07:09:14Z	rudolfsK	0 b	6
✓	F.CQ07JSIMGH5J8	Server.Utils.ImportCollection	2024-08-05T07:06:26Z	2024-08-05T07:06:26Z	rudolfsK	0 b	
✓	F.CQJL0BT3B78MU	Server.Utils.ImportCollection	2024-07-29T08:17:51Z	2024-07-29T08:17:54Z	karlisS	0 b	
✓	F.CQHPH32CFOK24	Server.Utils.CreateCollector	2024-07-26T12:37:32Z	2024-07-26T12:37:33Z	karlisS	56 Mb	
✓	F.CQHP75B834VHE	Server.Utils.CreateCollector	2024-07-26T12:16:21Z	2024-07-26T12:16:22Z	karlisS	56 Mb	

10 25 30 50 Showing 1 to 17 of 17 « 0 »

Timestamp	Level	message
2024-08-21T10:50:18Z	DEFAULT	Running query on behalf of user rudolfsK
2024-08-21T10:50:18Z	DEFAULT	<green>Starting</> query execution.
2024-08-21T10:50:18Z	DEFAULT	Searching for a client id with hostname 'contaboserver-21-08'
2024-08-21T10:50:18Z	DEFAULT	Creating a new client with id 'C.eef94f7989c7f87a'
2024-08-21T10:50:18Z	DEFAULT	import_collection: Copying /uploads/auto/etc/cron.monthly/%2Eplaceholder to /clients/C.eef94f7989c7f87a/collections/F.CR26B6ASPM6OE/uploads/auto/etc/cron.monthly/.placeholder
2024-08-21T10:50:18Z	DEFAULT	import_collection: Copying /uploads/auto/root/%2Ebashrc to /clients/C.eef94f7989c7f87a/collections/F.CR26B6ASPM6OE/uploads/auto/root/.bashrc
2024-08-21T10:50:18Z	DEFAULT	import_collection: Copying /uploads/auto/etc/cron.weekly/%2Eplaceholder to /clients/C.eef94f7989c7f87a/collections/F.CR26B6ASPM6OE/uploads/auto/etc/cron.weekly/.placeholder
2024-08-21T10:50:18Z	DEFAULT	import_collection: Copying /uploads/auto/etc/bash.bashrc to /clients/C.eef94f7989c7f87a/collections/F.CR26B6ASPM6OE/uploads/auto/etc/bash.bashrc
2024-08-21T10:50:18Z	DEFAULT	import_collection: Copying /uploads/auto/etc/cron.weekly/man-db to /clients/C.eef94f7989c7f87a/collections/F.CR26B6ASPM6OE/uploads/auto/etc/cron.weekly/man-db
2024-08-21T10:50:18Z	DEFAULT	import_collection: Copying /uploads/auto/etc/cron.hourly/%2Eplaceholder to /clients/C.eef94f7989c7f87a/collections/F.CR26B6ASPM6OE/uploads/auto/etc/cron.hourly/.placeholder

10 25 30 50 Showing 1 to 10 of 871 « 0 1 2 3 4 »