Velociraptor

Digging Deeper in Linux!

Velociraptor

- An advanced, open source endpoint visibility tool.
- Open architecture that is flexible and adaptable to new requirements.
- Implements advanced digital forensics techniques in the form of “Artifacts”
- Automates the boring stuff
- Scales to your entire endpoint fleet !

Who am I?

Dr Michael Cohen

- Experienced digital forensic software developer.
- Developer of foundation forensic tools including Volatility and Rekall.
- Former lead developer of Grr Rapid Response at Google Inc.

What will you need today?

A Linux computer or virtual machine, with admin access.

A copy of Velociraptor from our official release page:

https://github.com/Velocidex/velociraptor/releases

A hunting frame of mind.

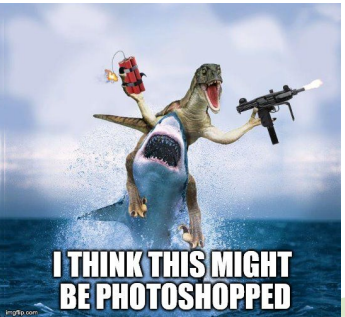
Velociraptor supports Windows, Linux and MacOS but today we will focus on **Linux!**

What is Velociraptor?

Velociraptor is a unique DFIR tool, giving *you* power and flexibility through the Velociraptor Query Language (VQL)

VQL is used for everything:

- ❏ Collecting information from endpoints (also called *clients*)
- ❏ Controlling monitoring and response on endpoints
- ❏ Controlling and managing the Velociraptor server.



Velociraptor overview

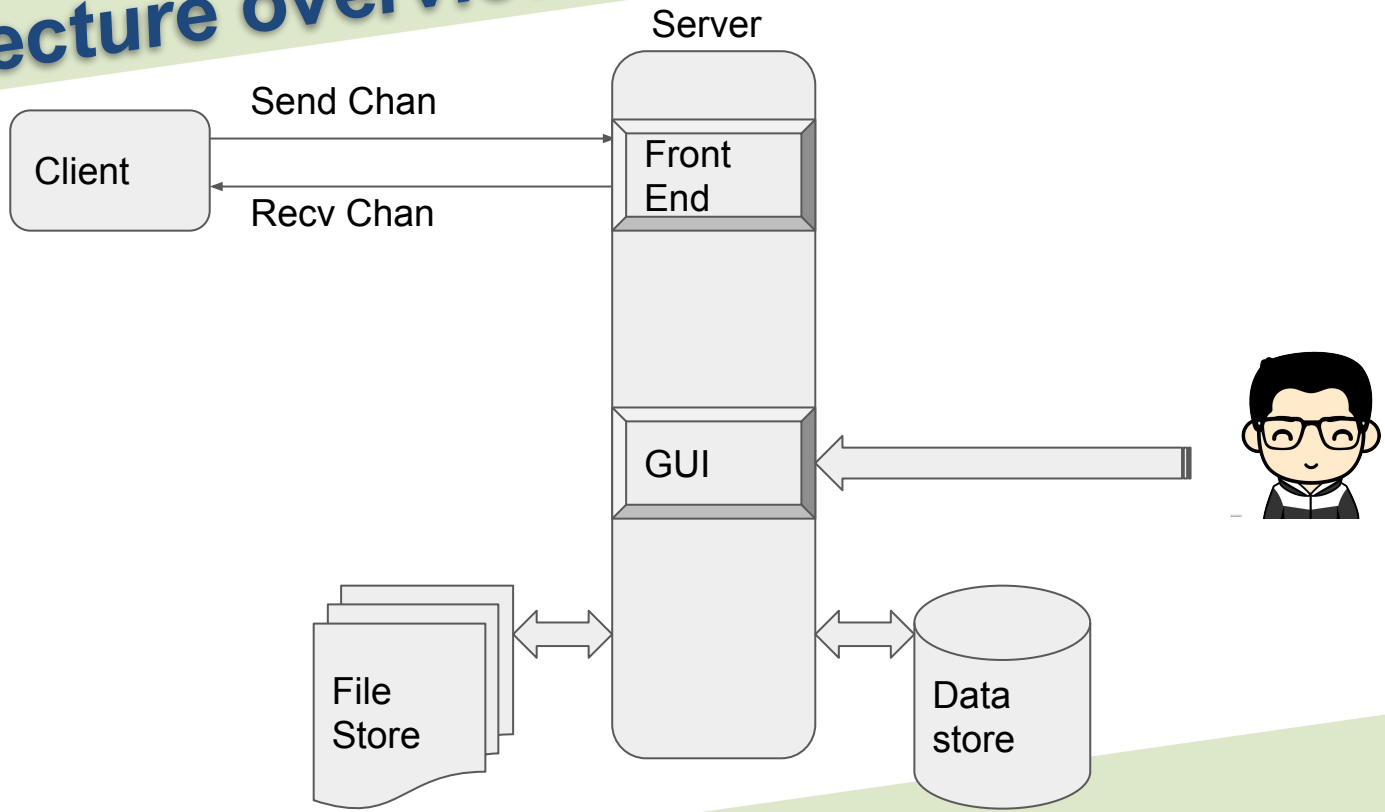
Everything uses the same binary - both clients and server.

- The server is controlled via the server configuration file.
- The client is controlled via the client configuration file.

In this lab, we run the server ***and*** client on the same machine.

In real cases, we typically deploy a Velociraptor server in the cloud.

Architecture overview



Installing Velociraptor

Download the latest Linux binary from our releases page:

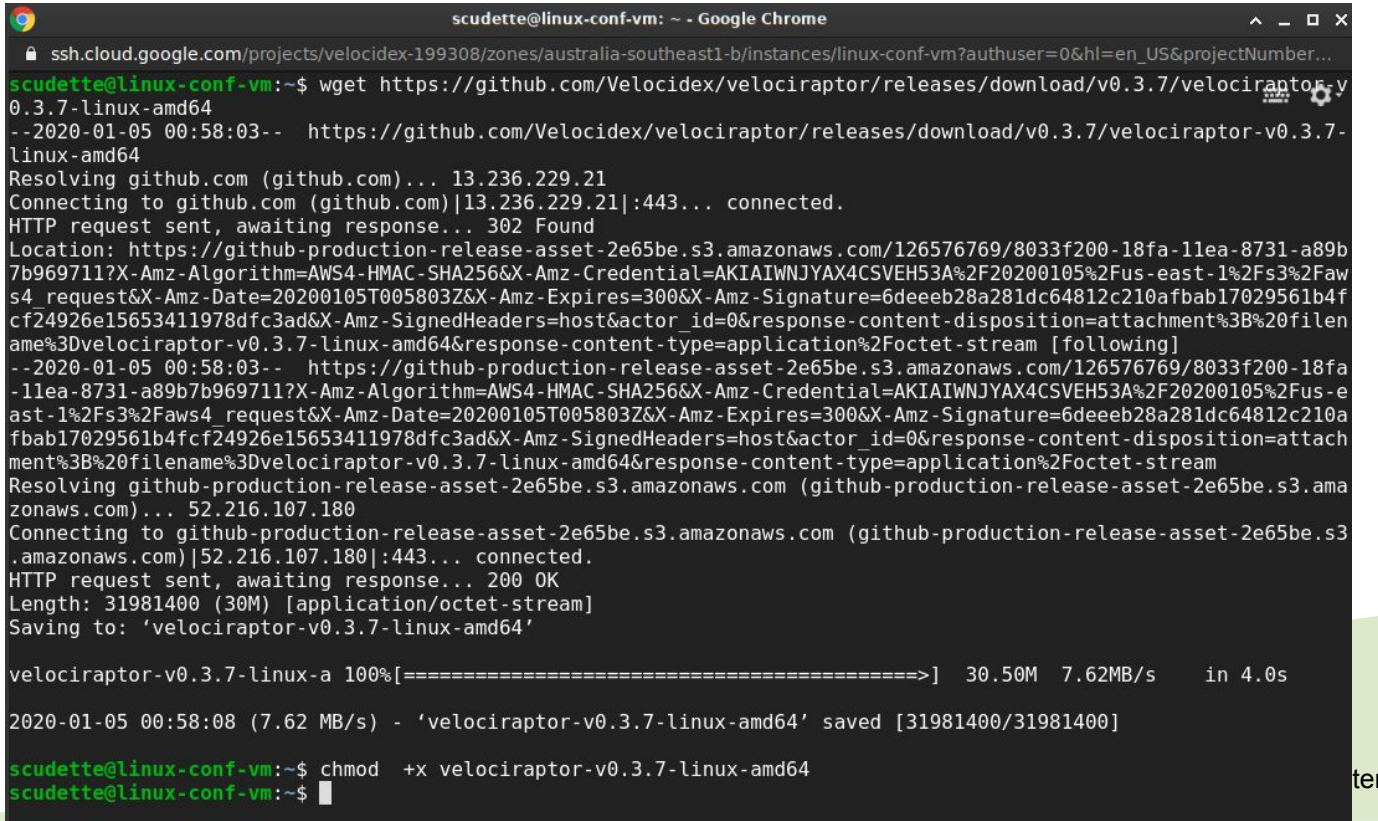
https://github.com/Velocidex/velociraptor/releases

It is just an ELF binary that should work on all Linux distributions later than about 2018.

```
wget https://github.com/Velocidex/velociraptor/releases/download/v0.3.8/velociraptor-v0.3.8-linux-amd64

chmod +x velociraptor-v0.3.8-linux-amd64
```

Download Velociraptor



terprises

Configuring Velociraptor

Everything is controlled by a pair of configuration files.

The configuration files contain key data, making them unique (and secure) to your deployment.

The server configuration file contains private keys - *make sure to secure it!*

Genering new configuration files is easy:

```
$ velociraptor config generate -i
```


```
scudette@linux-conf-vm:~$ ./velociraptor-v0.3.7-linux-amd64 config generate -i
?
Welcome to the Velociraptor configuration generator
--------------------------------------------------

I will be creating a new deployment configuration for you. I will
begin by identifying what type of deployment you need.

 Self Signed SSL
Generating keys please wait....
? Enter the frontend port to listen on. 8000
? What is the public DNS name of the Frontend (e.g. www.example.com): localhost
? Path to the datastore directory. /tmp
? Path to the logs directory. /tmp
? GUI Username or email address to authorize (empty to end): mic
? GUI Username or email address to authorize (empty to end):
? Where should i write the server config file? server.config.yaml
? Where should i write the client config file? client.config.yaml
scudette@linux-conf-vm:~$
```

Starting the server

The same binary acts as a server or client depending on configuration options.

The previous step generated two files:

```
client.config.yaml
server.config.yaml
```

Start the Velociraptor server and frontend:

```
velociraptor --config server.config.yaml frontend -v
```


Starting the server

```
scudette@linux-conf-vm:~$ ./velociraptor-v0.3.7-linux-amd64  --config server.config.yaml frontend -v
[INFO] 2020-01-05T01:10:36Z Starting Frontend. {"build_time":"2019-12-07T12:56:30+10:00","commit":"ff4d5f7","version":"0.3.7"}
[INFO] 2020-01-05T01:10:36Z Loaded 137 built in artifacts
[INFO] 2020-01-05T01:10:36Z Increased open file limit to 999999
[INFO] 2020-01-05T01:10:36Z Launched Prometheus monitoring server on 127.0.0.1:8003
[INFO] 2020-01-05T01:10:36Z Frontend is ready to handle client TLS requests at 0.0.0.0:8000
[INFO] 2020-01-05T01:10:36Z Starting hunt manager.
[INFO] 2020-01-05T01:10:36Z Launched gRPC API server on 127.0.0.1:8001
[INFO] 2020-01-05T01:10:36Z GUI is ready to handle TLS requests {"listenAddr":"127.0.0.1:8889"}
[INFO] 2020-01-05T01:10:36Z Starting Hunt Dispatcher Service.
[INFO] 2020-01-05T01:10:36Z Starting Stats Collector Service.
[INFO] 2020-01-05T01:10:36Z Starting Server Monitoring Service
[INFO] 2020-01-05T01:10:36Z Starting Server Artifact Runner Service
[INFO] 2020-01-05T01:10:36Z Starting Client Monitoring Service
[INFO] 2020-01-05T01:10:36Z Collecting Client Monitoring Artifact: Generic.Client.Stats
[INFO] 2020-01-05T01:10:36Z Collecting Client Monitoring Artifact: Windows.Events.ProcessCreation
[INFO] 2020-01-05T01:10:36Z Initial user mic not present, creating
[INFO] 2020-01-05T01:10:36Z Starting VFS writing service.
[INFO] 2020-01-05T01:10:36Z Collecting Server Event Artifact: Server.Monitor.Health/Prometheus
[INFO] 2020-01-05T01:10:36Z Starting interrogation service.
```

Test that the GUI works

Connect to the GUI address mentioned previously:

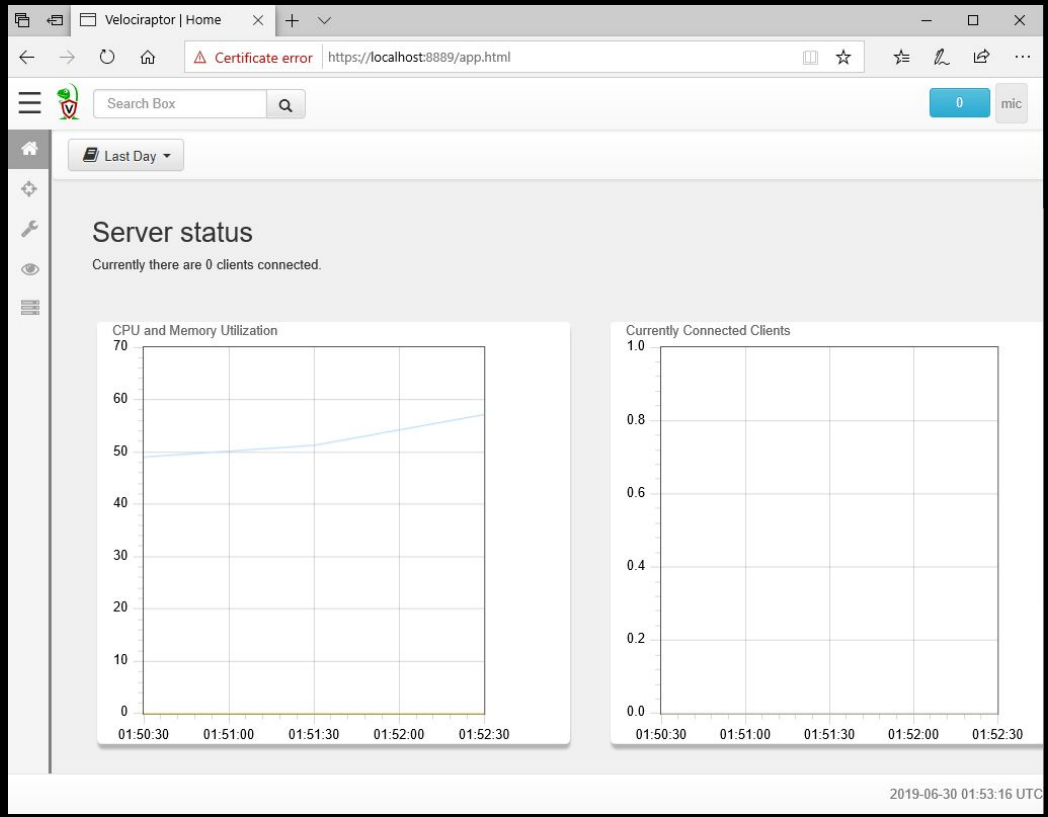
https://localhost:8889/

Note the certificate error - *this is OK*. It's because we chose self-signed SSL mode. You can click through the warning for now.

In real deployments we use proper SSL certificates.

Your Velociraptor server is ready.

Now let's configure some clients.



Creating a debian package

To permanently install the client and server we need to create debian packages

```
velociraptor --config server.config.yaml debian server
velociraptor --config server.config.yaml debian client
```


Creating a debian package

scudette@linux-conf-vm (10.152.0.3) - byobu - Google Chrome

ssh.cloud.google.com/projects/velocidex-199308/zones/australia-southeast1-b/instances/linux-conf-vm?authuser=0&hl=en_US&projectN...

```
scudette@linux-conf-vm:~$ ./velociraptor-v0.3.7-linux-amd64 --config server.config.yaml debian server
scudette@linux-conf-vm:~$ ./velociraptor-v0.3.7-linux-amd64 --config server.config.yaml debian client
scudette@linux-conf-vm:~$ ls -l *.deb
-rw-r--r-- 1 scudette scudette 13914570 Jan  5 01:57 velociraptor_0.3.7_client.deb
-rw-r--r-- 1 scudette scudette 13923280 Jan  5 01:56 velociraptor_0.3.7_server.deb
scudette@linux-conf-vm:~$ sudo dpkg -i velociraptor_0.3.7_server.deb
(Reading database ... 38225 files and directories currently installed.)
Preparing to unpack velociraptor_0.3.7_server.deb ...
Removed /etc/systemd/system/multi-user.target.wants/velociraptor_server.service.
Unpacking velociraptor-server (0.3.7) over (0.3.7) ...
Setting up velociraptor-server (0.3.7) ...
Created symlink /etc/systemd/system/multi-user.target.wants/velociraptor_server.service → /etc/systemd/syst
em/velociraptor_server.service.
scudette@linux-conf-vm:~$ sudo dpkg -i velociraptor_0.3.7_client.deb
Selecting previously unselected package velociraptor-client.
(Reading database ... 38225 files and directories currently installed.)
Preparing to unpack velociraptor_0.3.7_client.deb ...
Unpacking velociraptor-client (0.3.7) ...
Setting up velociraptor-client (0.3.7) ...
Created symlink /etc/systemd/system/multi-user.target.wants/velociraptor_client.service → /etc/systemd/syst
em/velociraptor_client.service.
scudette@linux-conf-vm:~$
```

The Dashboard

The **Dashboard** shows the current state of the installation:

- How many clients are connected
- Current CPU load and memory footprint on the server.

When running hunts or intensive processing, memory and CPU requirements will increase but not too much.

You can customize the dashboard - it's also just an artifact.

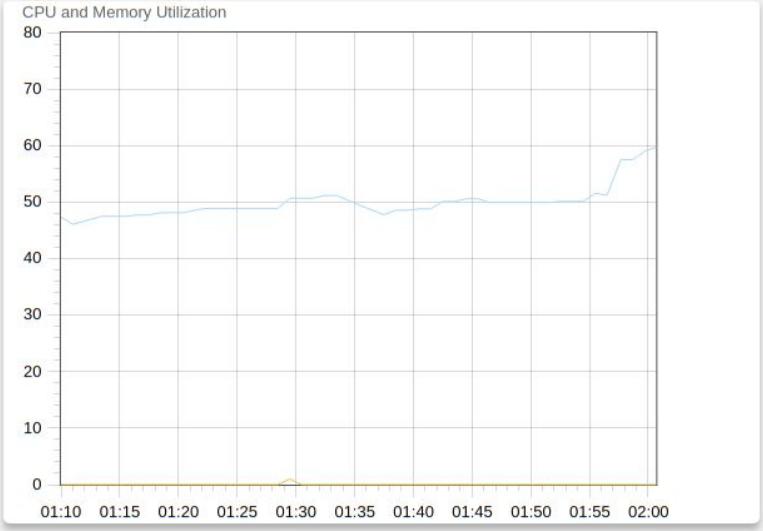
Search Box

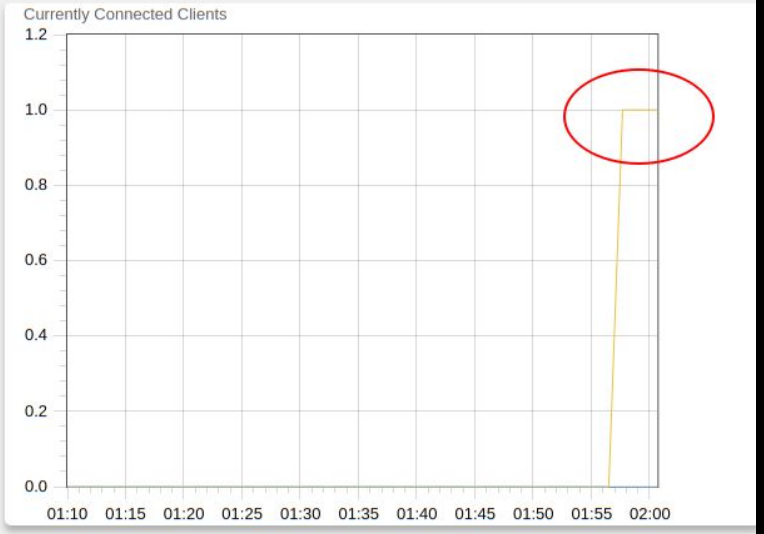
0

Last Day

Server status

Currently there are 1 clients connected.





Users

Show 10 entries

Search:

Name	ReadOnly
mic	false

Clients have a persistent connection to the server.

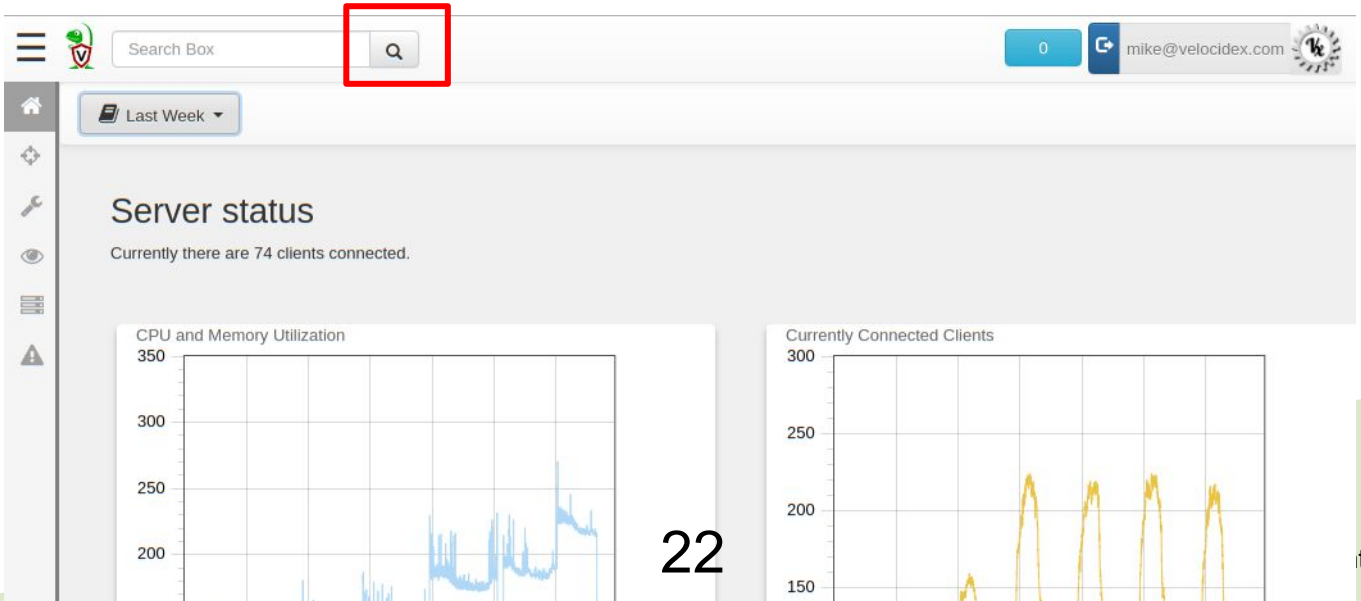
They’re ready to receive your commands.

Interactively investigate individual clients

Searching for a client

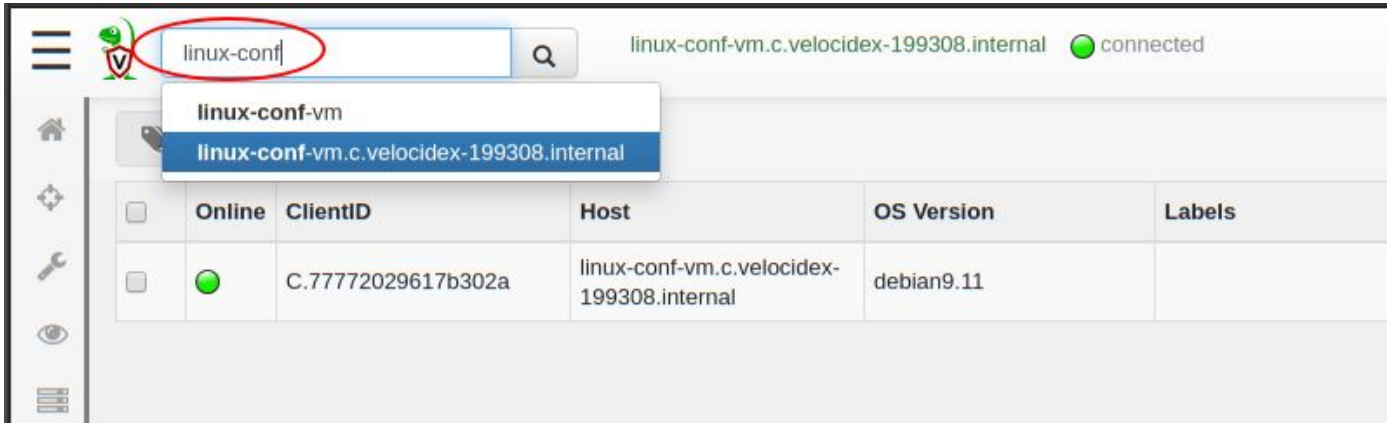
To work with a specific client we need to search for it.

Press the **Search** icon to see all the clients



terprises

Search for clients hostname, label or client ID.



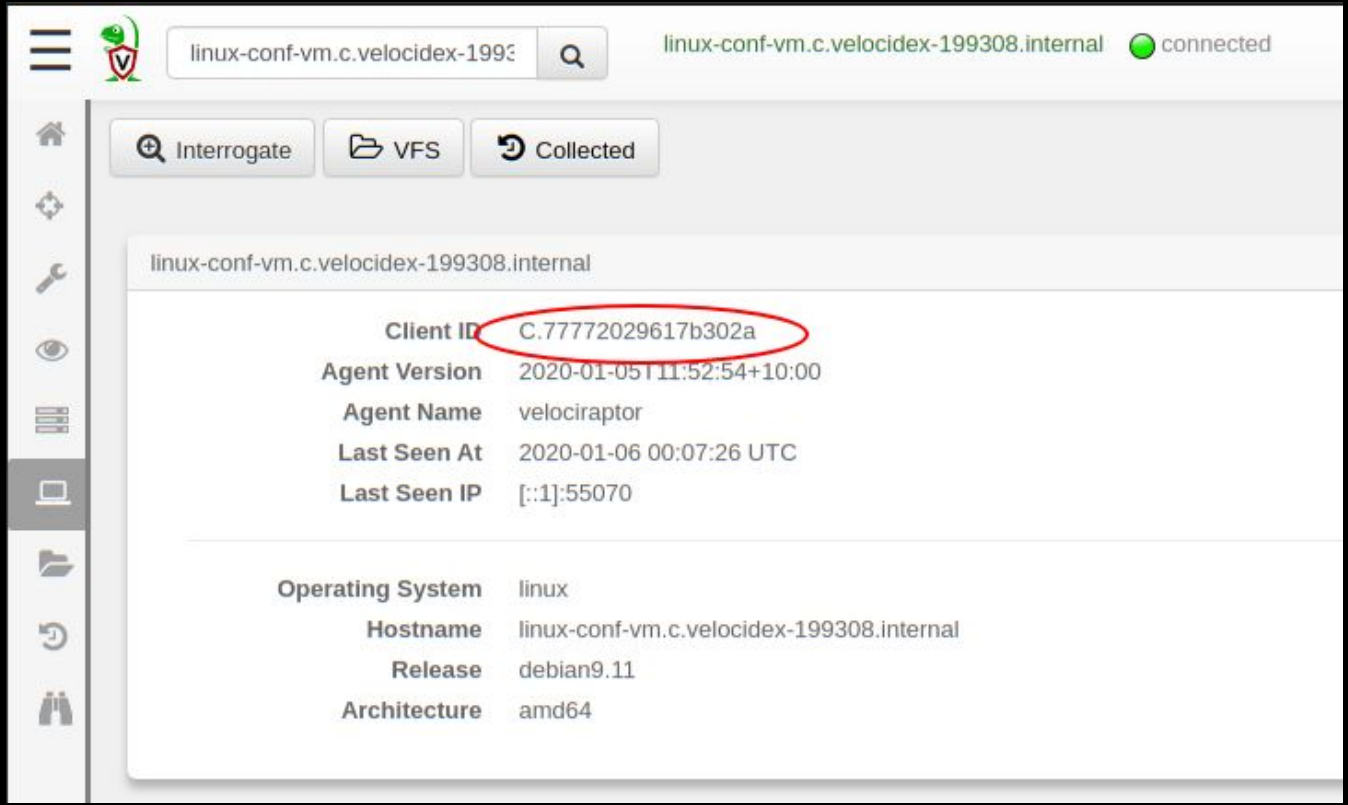
Client overview

The server collects some high level information about each endpoint.

Click **VQL Drilldown** to see more detailed information:

- ❑ Client version
- ❑ Client footprint (memory and CPU)

You can customize the information collected and shown by editing the **Generic.Client.Info** artifact.



Search Box

linux-conf-vm.c.velocidex-199308.internal connected

0 mic

Interrogate VFS Collected

Overview VQL Drilldown

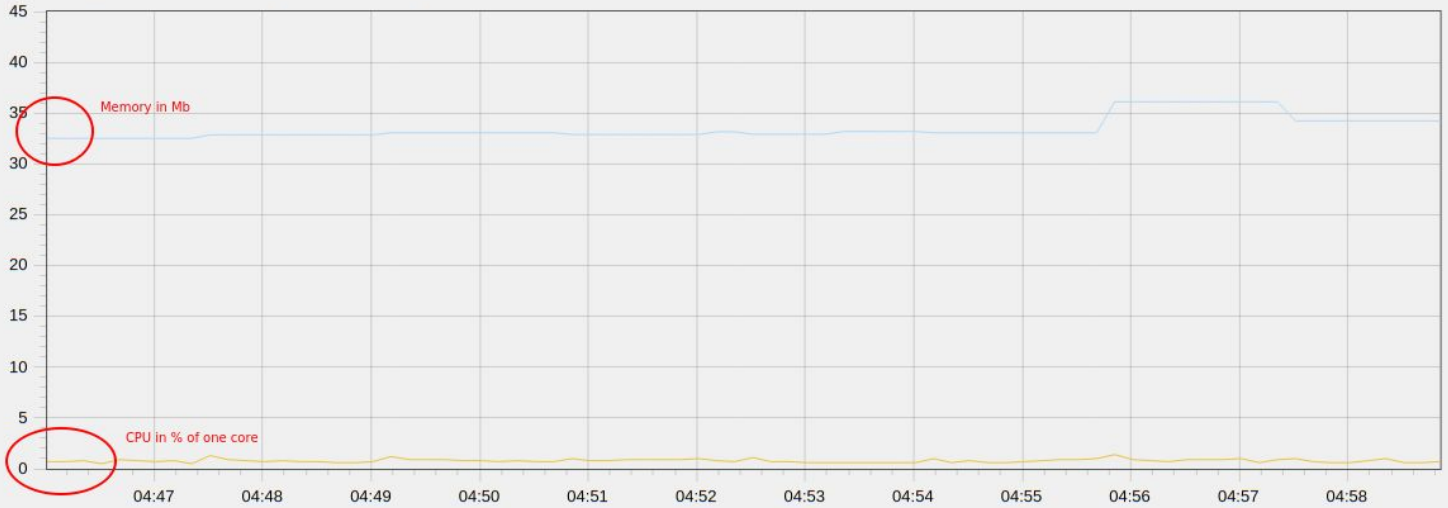
linux-conf-vm.c.velocidex-199308.internal (C.77772029617b302a) @ 2020-01-05 01:55:39 +0000 UTC

Show 10 entries Search:

Name	BuildTime	Labels	Hostname	OS	Architecture	Platform	PlatformVersion	KernelVersion	Fqdn
velociraptor	2020-01-05T11:52:54+10:00	-	linux-conf-vm	linux	amd64	debian	9.11	4.9.0-11-amd64	linux-conf-vm.c.velocidex-199308.internal

Showing 1 to 1 of 1 entries Previous 1 Next

Memory and CPU footprint over the past 24 hours

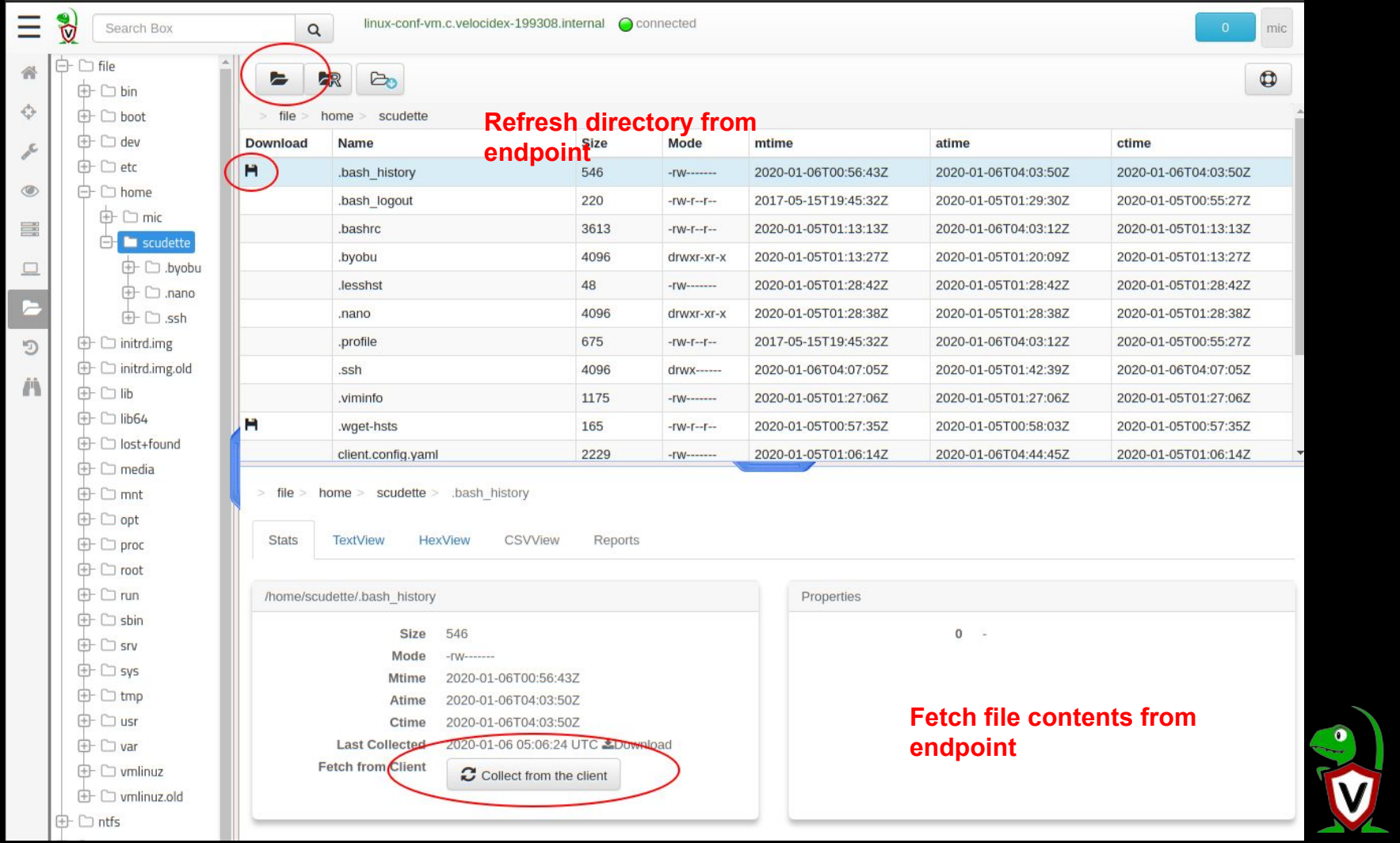


The Virtual File System (VFS)

The VFS visualizes some server-side information we collect about the clients.

Top level corresponds to the type of information we collect:

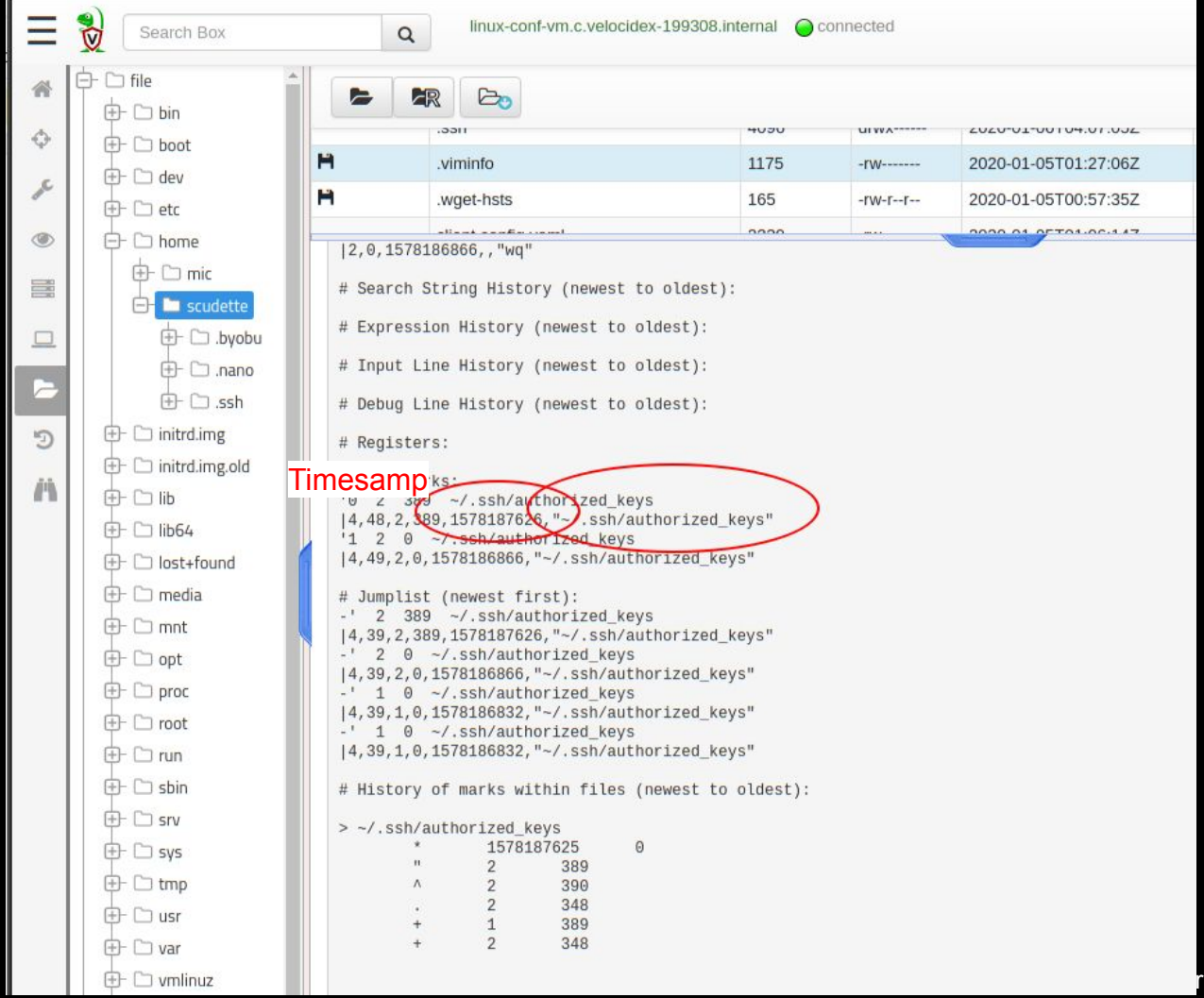
- **File** - Access the file system using the filesystem API
- **NTFS** - Access the file system using raw NTFS parsing (Windows Only)
- **Registry** - Access the Windows Registry using the Registry API (Windows Only)
- **Artifacts** - A view of all artifacts collected from the client.


Search Box
linux-conf-vm.c.velocidex-199308.internal
connected
0
mic
file
bin
boot
dev
etc
home
mic
scudette
.byobu
.nano
.ssh
initrd.img
initrd.img.old
lib
lib64
lost+found
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
vmlinuz
vmlinuz.old
ntfs
file > home > scudette
Refresh directory from endpoint
Download
Name
Size
Mode
mtime
atime
ctime
.bash_history 546 -rw------- 2020-01-06T00:56:43Z 2020-01-06T04:03:50Z 2020-01-06T04:03:50Z
.bash_logout 220 -rw-r--r-- 2017-05-15T19:45:32Z 2020-01-05T01:29:30Z 2020-01-05T00:55:27Z
.bashrc 3613 -rw-r--r-- 2020-01-05T01:13:13Z 2020-01-06T04:03:12Z 2020-01-05T01:13:13Z
.byobu 4096 drwxr-xr-x 2020-01-05T01:13:27Z 2020-01-05T01:20:09Z 2020-01-05T01:13:27Z
.lesshst 48 -rw------- 2020-01-05T01:28:42Z 2020-01-05T01:28:42Z 2020-01-05T01:28:42Z
.nano 4096 drwxr-xr-x 2020-01-05T01:28:38Z 2020-01-05T01:28:38Z 2020-01-05T01:28:38Z
.profile 675 -rw-r--r-- 2017-05-15T19:45:32Z 2020-01-06T04:03:12Z 2020-01-05T00:55:27Z
.ssh 4096 drwx------ 2020-01-06T04:07:05Z 2020-01-05T01:42:39Z 2020-01-06T04:07:05Z
.viminfo 1175 -rw------- 2020-01-05T01:27:06Z 2020-01-05T01:27:06Z 2020-01-05T01:27:06Z
.wget-hsts 165 -rw-r--r-- 2020-01-05T00:57:35Z 2020-01-05T00:58:03Z 2020-01-05T00:57:35Z
client.config.yaml 2229 -rw------- 2020-01-05T01:06:14Z 2020-01-06T04:44:45Z 2020-01-05T01:06:14Z
file > home > scudette > .bash_history
Stats
TextView
HexView
CSVView
Reports
/home/scudette/.bash_history
Size 546
Mode -rw-------
Mtime 2020-01-06T00:56:43Z
Atime 2020-01-06T04:03:50Z
Ctime 2020-01-06T04:03:50Z
Last Collected 2020-01-06 05:06:24 UTC Download
Fetch from Client
Collect from the client
Properties
0 -
Fetch file contents from endpoint

Exercise: Determine user activity

Task: We suspect a user account had been compromised. Determine what the attacker had done.

- Useful artifacts include
  - Bash history
  - Less history
  - VIM history


Search Box
linux-conf-vm.c.velocidex-199308.internal
connected
file
bin
boot
dev
etc
home
mic
scudette
.byobu
.nano
.ssh
initrd.img
initrd.img.old
lib
lib64
lost+found
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
vmlinuz
.viminfo 1175 -rw------- 2020-01-05T01:27:06Z
.wget-hsts 165 -rw-r--r-- 2020-01-05T00:57:35Z
|2,0,1578186866,,"wq"
Search String History (newest to oldest):
Expression History (newest to oldest):
Input Line History (newest to oldest):
Debug Line History (newest to oldest):
Registers:
Timesamp
|4,48,2,389,1578187626,"~/.ssh/authorized_keys"
'1 2 0 ~/.ssh/authorized_keys
|4,49,2,0,1578186866,"~/.ssh/authorized_keys"
Jumplist (newest first):
-' 2 389 ~/.ssh/authorized_keys
|4,39,2,389,1578187626,"~/.ssh/authorized_keys"
-' 2 0 ~/.ssh/authorized_keys
|4,39,2,0,1578186866,"~/.ssh/authorized_keys"
-' 1 0 ~/.ssh/authorized_keys
|4,39,1,0,1578186832,"~/.ssh/authorized_keys"
-' 1 0 ~/.ssh/authorized_keys
|4,39,1,0,1578186832,"~/.ssh/authorized_keys"
History of marks within files (newest to oldest):
> ~/.ssh/authorized_keys
* 1578187625 0
" 2 389
^ 2 390
. 2 348
+ 1 389
+ 2 348

Use Velociraptor artifacts to automate everything

Automate everything!

We can collect information about *many* things in DFIR cases:

- ❑ Registry keys, files, WMI queries, sqlite databases …

But we really just want to answer specific questions:

- ❑ What program did the attacker run?
- ❑ What files were downloaded?
- ❑ Did the user connect to a known malicious C&C?
- ❑ Did a particular file exist on a client?

Velociraptor uses expert knowledge to find the evidence

A key objective of Velociraptor is encapsulating DFIR knowledge into the platform, so you don't need to be a DFIR expert.

- ❑ We have high level questions to answer
- ❑ We know where to look for evidence of user / system activities

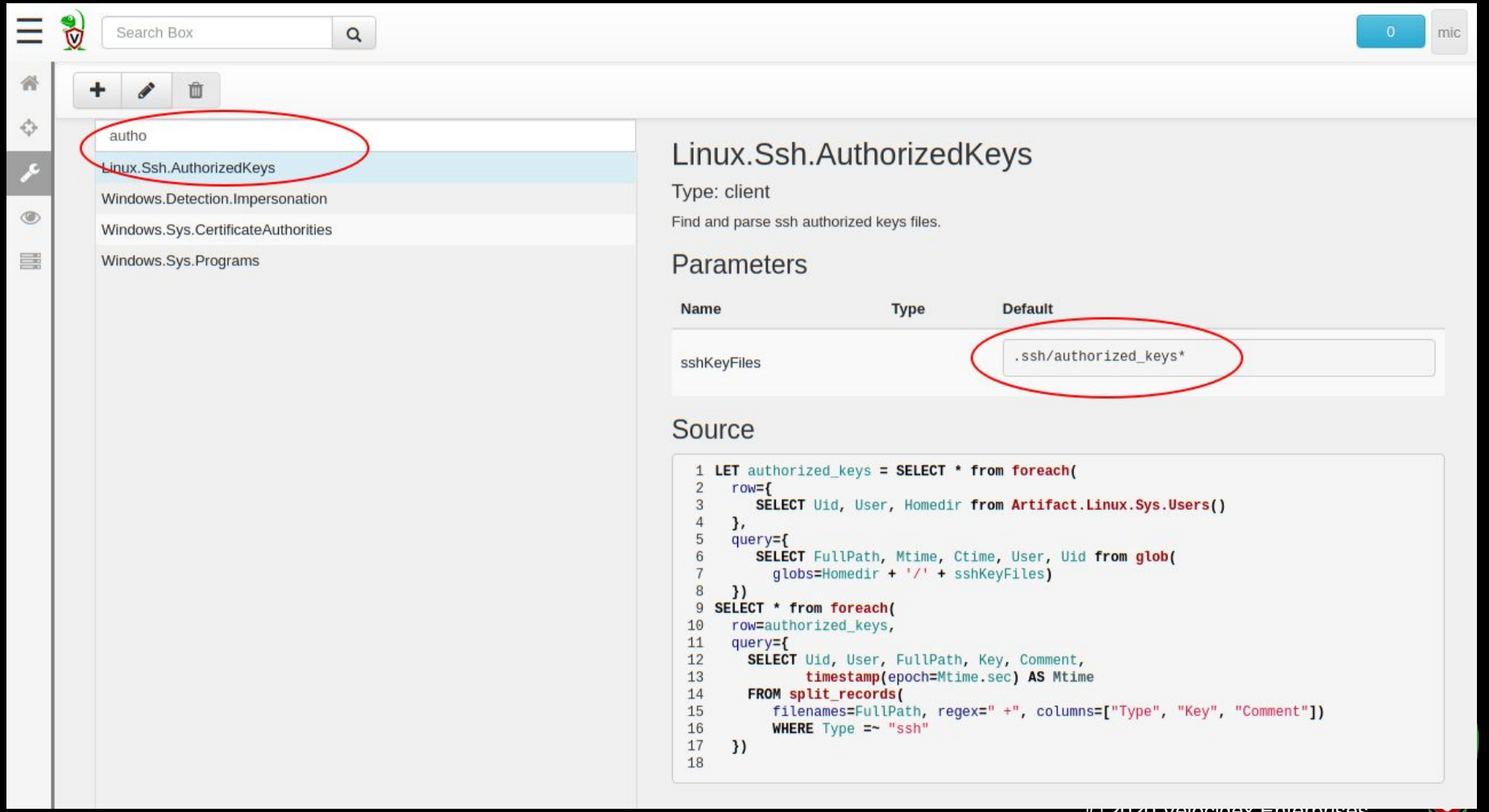
We build artifacts to collect and analyze the evidence in order to answer our investigative questions.

Velociraptor's superpower: user specified artifacts

An artifact is a YAML file …

- ❏ (therefore user-readable, shareable and editable)
- ❏ … that answers a question …
- ❏ … by collecting data from the endpoint …
- ❏ … and reporting on this data in a human readable way.

Artifacts encode expert knowledge into human reusable components.



Exercise: Collect all users' authorized keys

Select **Collected Artifacts** to view all artifacts previously collected.

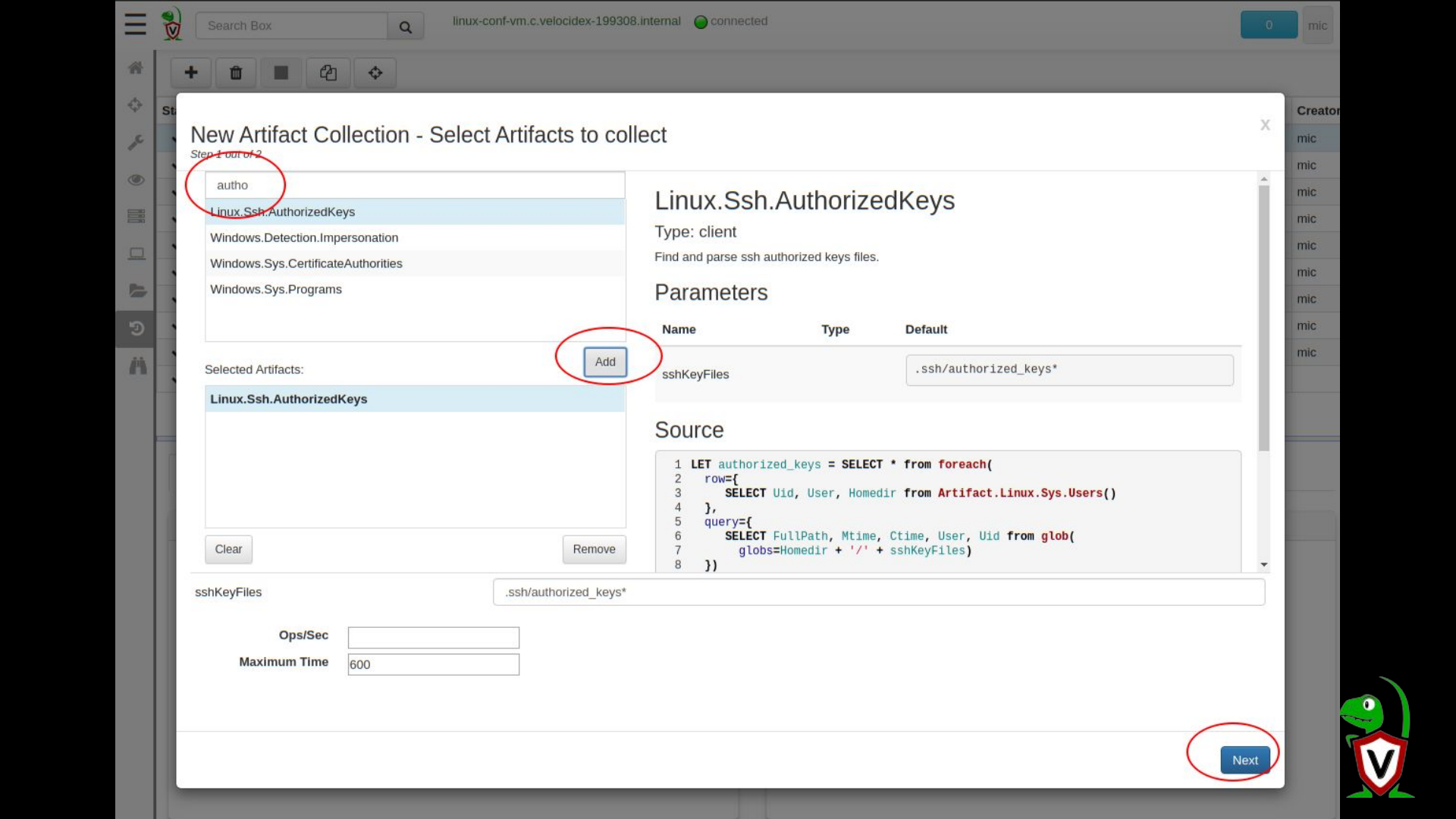
Click **Collect More Artifacts** to open the **New Artifact Wizard**.

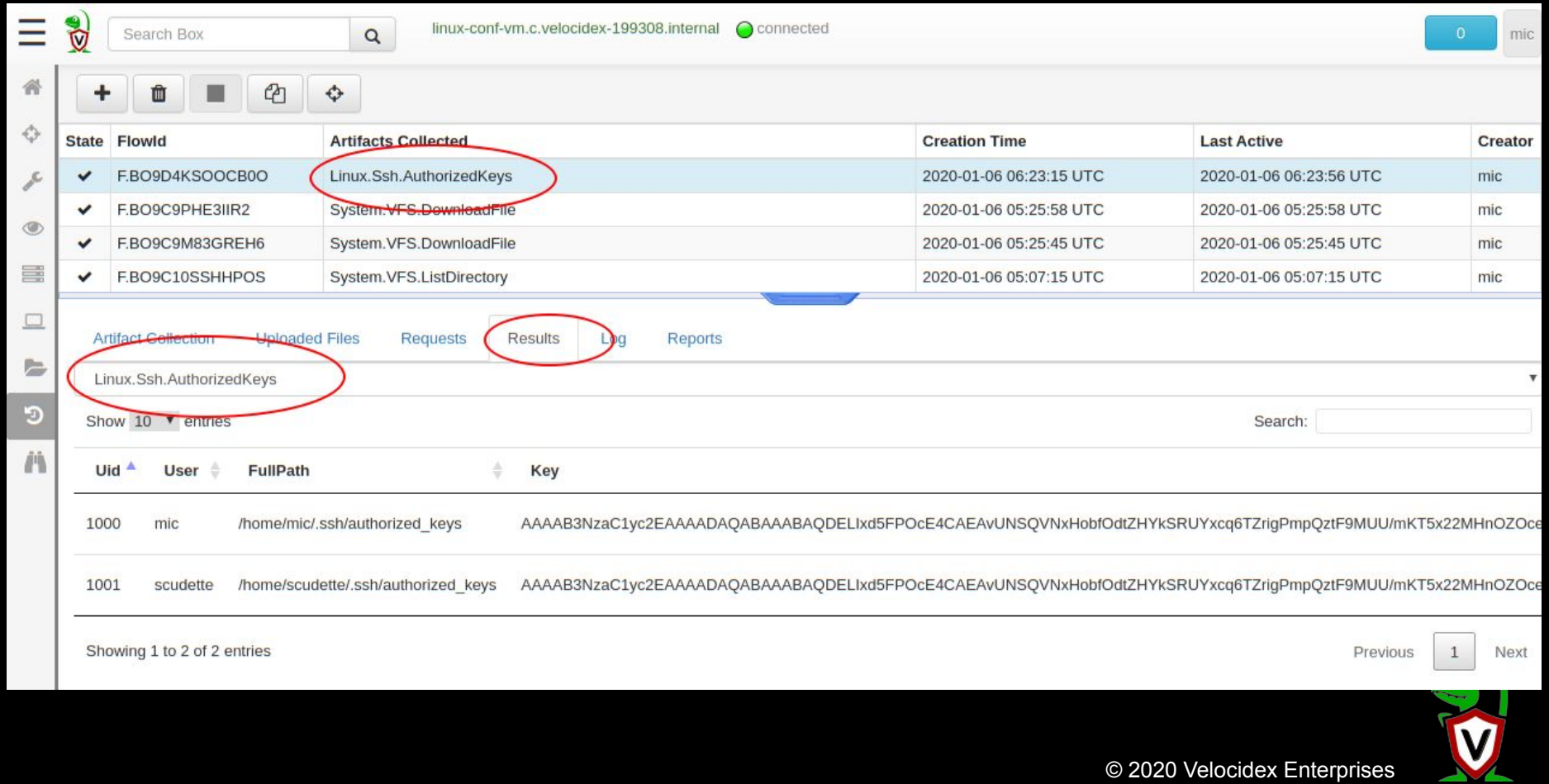
Search for an artifact that fetches authorized keys.

Click **Add** to add the artifact to the list for collection.

Click **Next** to start the collection.

This allows us to determine which keys provide access to which user account


New Artifact Collection - Select Artifacts to collect
Step 1 out of 2
autho
Linux.Ssh.AuthorizedKeys
Windows.Detection.Impersonation
Windows.Sys.CertificateAuthorities
Windows.Sys.Programs
Add
Selected Artifacts:
Linux.Ssh.AuthorizedKeys
Clear
Remove
Linux.Ssh.AuthorizedKeys
Type: client
Find and parse ssh authorized keys files.
Parameters
Name
Type
Default
sshKeyFiles
.ssh/authorized_keys*
Source
LET authorized_keys = SELECT * from foreach(
row={
SELECT Uid, User, Homedir from Artifact.Linux.Sys.Users()
},
query={
SELECT FullPath, Mtime, Ctime, User, Uid from glob(
globs=Homedir + '/' + sshKeyFiles)
})
sshKeyFiles
.ssh/authorized_keys*
Ops/Sec
Maximum Time
600
Next

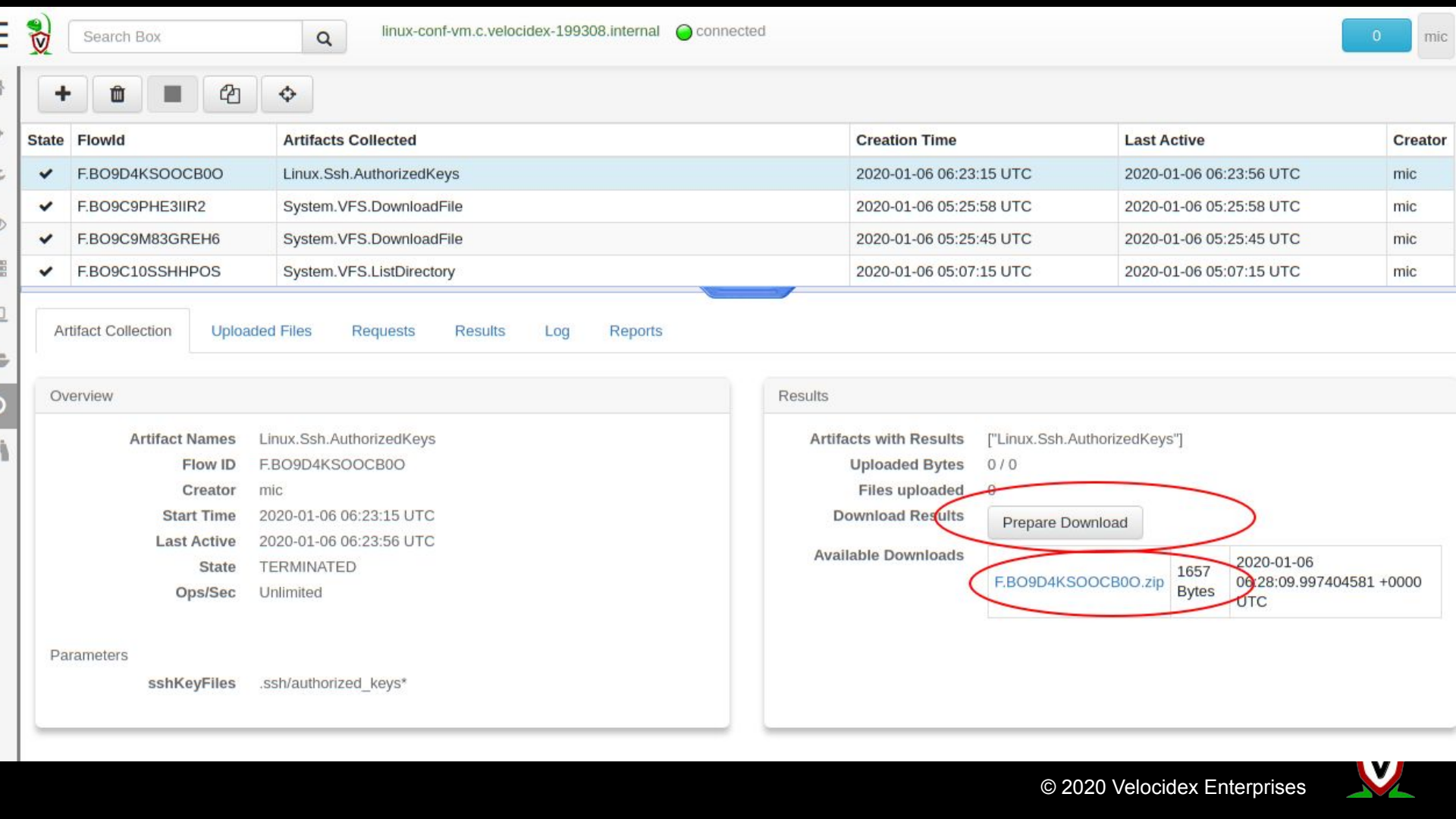


Get the collected data

The artifact is running a VQL query - returning one row per key.

Click **Prepare Download** to download the results of this artifact collection through your web browser (see next slide).

The result is a ZIP file with any collected files and a CSV file of the collection results.



State	FlowId	Artifacts Collected	Creation Time	Last Active	Creator
✔	F.BO9D4KSOOCB0O	Linux.Ssh.AuthorizedKeys	2020-01-06 06:23:15 UTC	2020-01-06 06:23:56 UTC	mic
✔	F.BO9C9PHE3IIR2	System.VFS.DownloadFile	2020-01-06 05:25:58 UTC	2020-01-06 05:25:58 UTC	mic
✔	F.BO9C9M83GREH6	System.VFS.DownloadFile	2020-01-06 05:25:45 UTC	2020-01-06 05:25:45 UTC	mic
✔	F.BO9C10SSHHPOS	System.VFS.ListDirectory	2020-01-06 05:07:15 UTC	2020-01-06 05:07:15 UTC	mic

Artifact Collection | Uploaded Files | Requests | Results | Log | Reports

Overview

Artifact Names Linux.Ssh.AuthorizedKeys
Flow ID F.BO9D4KSOOCB0O
Creator mic
Start Time 2020-01-06 06:23:15 UTC
Last Active 2020-01-06 06:23:56 UTC
State TERMINATED
Ops/Sec Unlimited

Parameters

sshKeyFiles .ssh/authorized_keys*

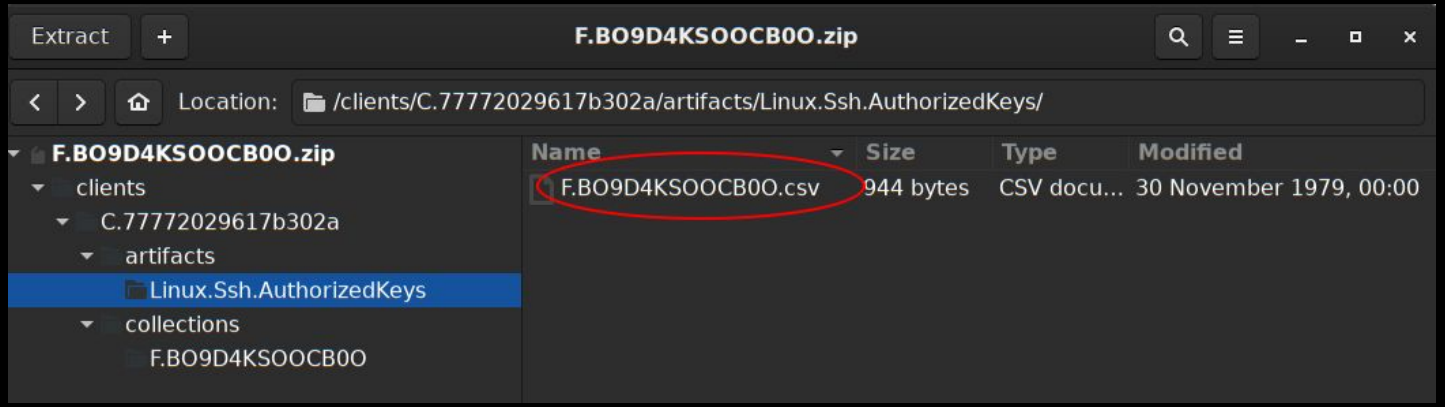
Results

Artifacts with Results ["Linux.Ssh.AuthorizedKeys"]
Uploaded Bytes 0 / 0
Files uploaded 0
Download Results Prepare Download

Available Downloads

F.BO9D4KSOOCB0O.zip	1657 Bytes	2020-01-06 06:28:09.997404581 +0000 UTC

The ZIP file contains a directory structure for each client with the collected artifacts stored in csv files.



Scenario - Admin is leaving

- Bob is an IT administrator with a lot of access.
- Bob just sent his resignation letter.
- Which machines of our 25,000 servers/laptops/cloud instances, does Bob have access to?
- What systems has Bob logged into in the last 2 months?

BTW …..

We don’t have SSO or centralized logging ¯_(ツ)_/¯

Query local SSH logs

SSH logins are normally recorded in /var/log/auth.log which is rotated periodically. e.g.:

Jan 5 09:56:55 DevBox sshd[1953]: Accepted password for mic from 192.168.0.5 port 36836 ssh2

We want to parse the logs for this specific message. We could write a regular expression

… but …

Grok is a de facto standard in the world of log parsing ….

Grokking authorization logs

Grok expressions are used to apply regex to log lines and capture into structured JSON dict

Jan 5 09:56:55 DevBox sshd[1953]: Accepted password for mic from 192.168.0.5 port 36836 ssh2

```
%{SYSLOGTIMESTAMP:Timestamp}
%{SYSLOGHOST:logsource} %{SYSLOGPROG}:
%{DATA:event} %{DATA:method} for (invalid
user )?%{DATA:user} from %{IPORHOST:ip}
port %{NUMBER:port} ssh2(:
%{GREEDYDATA:system.auth.ssh.signature})?
```

Linux.Syslog.SSHLogin

Type: client

Parses the auth logs to determine all SSH login attempts.

Parameters

Name	Type	Default
syslogAuthLogPath		/var/log/auth.log*
SSHGrok		%{SYSLOGTIMESTAMP:Timestamp} (?:%{SYSLOGFACILITY})?%{SYSLOGHOST:

Source

```
1  SELECT timestamp(string=Event.Timestamp) AS Time,
2         Event.IP AS IP,
3         Event.event AS Result,
4         Event.method AS Method,
5         Event.user AS AttemptedUser,
6         FullPath
7  FROM foreach(
8    row={
9      SELECT FullPath FROM glob(globs=syslogAuthLogPath)
10   }, query={
11     SELECT grok(grok=SSHGrok, data=Line) AS Event, FullPath
12     FROM parse_lines(filename=FullPath)
13     WHERE Event.program = "sshd"
14   })
15
```

New Artifact Collection - Select Artifacts to collect

Step 1 out of 2

syslog

Linux.Syslog.SSHLogin

Add

Selected Artifacts:

Linux.Syslog.SSHLogin

Clear

Remove

syslogAuthLogPath	/var/log/auth.log*
SSHGrok	%{SYSLOGTIMESTAMP:Timestamp} (?:%{SYSLOGFACILITY})?%{SYSL

Source

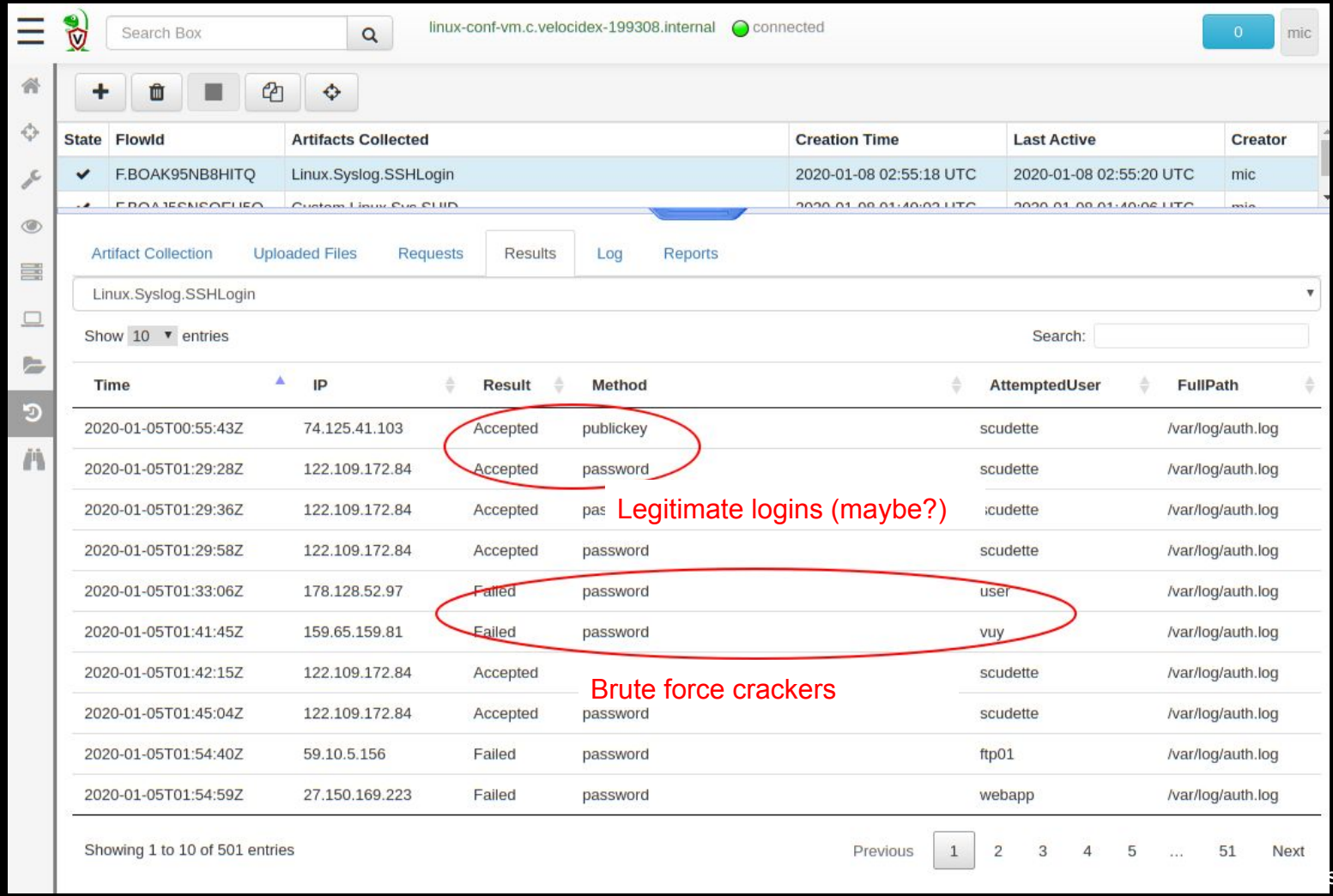
```
1  SELECT timestamp(string=Event.Timestamp) AS Time,
2         Event.IP AS IP,
3         Event.event AS Result,
4         Event.method AS Method,
5         Event.user AS AttemptedUser,
6         FullPath
7  FROM foreach(
8    row={
9        SELECT FullPath FROM glob(globs=syslogAuthLogPath)
10   }, query={
11       SELECT grok(grok=SSHGrok, data=Line) AS Event, FullPath
12       FROM parse_lines(filename=FullPath)
13       WHERE Event.program = "sshd"
14   })
15
```

syslogAuthLogPath	/var/log/auth.log*
SSHGrok	%{SYSLOGTIMESTAMP:Timestamp} (?:%{SYSLOGFACILITY})?%{SYSLOGHOST:logsource} %{SYSLOGPRO(

Ops/Sec

Maximum Time 600

Next


Search Box
linux-conf-vm.c.velocidex-199308.internal
connected
0
mic
State
FlowId
Artifacts Collected
Creation Time
Last Active
Creator
F.BOAK95NB8HITQ
Linux.Syslog.SSHLogin
2020-01-08 02:55:18 UTC
2020-01-08 02:55:20 UTC
mic
Artifact Collection
Uploaded Files
Requests
Results
Log
Reports
Linux.Syslog.SSHLogin
Show 10 entries
Search:
Time
IP
Result
Method
AttemptedUser
FullPath
2020-01-05T00:55:43Z 74.125.41.103 Accepted publickey scudette /var/log/auth.log
2020-01-05T01:29:28Z 122.109.172.84 Accepted password scudette /var/log/auth.log
2020-01-05T01:29:36Z 122.109.172.84 Accepted pas scudette /var/log/auth.log
2020-01-05T01:29:58Z 122.109.172.84 Accepted password scudette /var/log/auth.log
2020-01-05T01:33:06Z 178.128.52.97 Failed password user /var/log/auth.log
2020-01-05T01:41:45Z 159.65.159.81 Failed password vuy /var/log/auth.log
2020-01-05T01:42:15Z 122.109.172.84 Accepted scudette /var/log/auth.log
2020-01-05T01:45:04Z 122.109.172.84 Accepted password scudette /var/log/auth.log
2020-01-05T01:54:40Z 59.10.5.156 Failed password ftp01 /var/log/auth.log
2020-01-05T01:54:59Z 27.150.169.223 Failed password webapp /var/log/auth.log
Legitimate logins (maybe?)
Brute force crackers
Showing 1 to 10 of 501 entries
Previous
1
2
3
4
5
...
51
Next

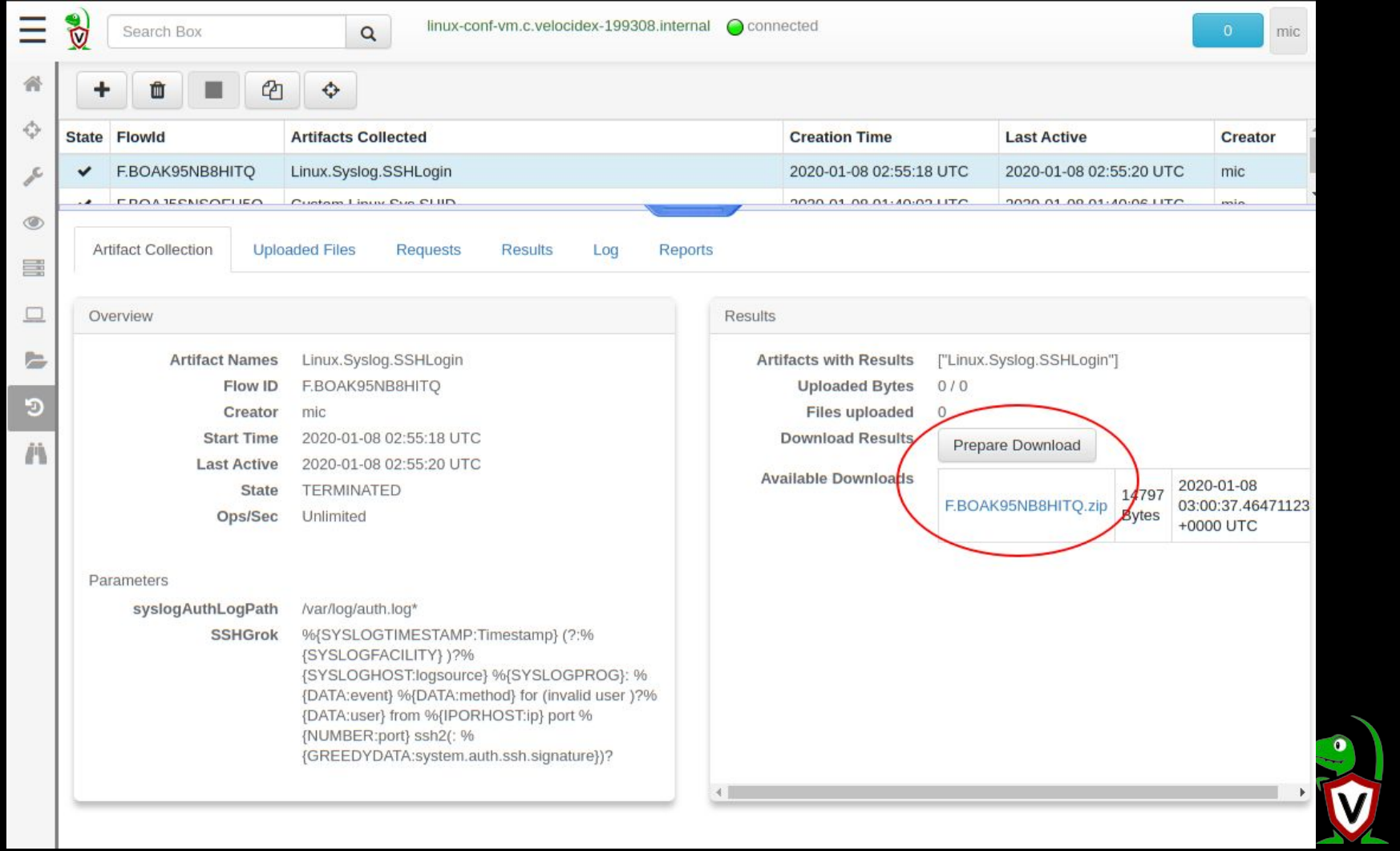
Offline analysis

Velociraptors job is just to collect and preserve evidence:

- ❏ Collect files to the server
- ❏ Run queries on the endpoint and store result sets on the server

Velociraptor **does not** index or analyze the results of the queries!

You can export data as Zip files containing the CSV or JSON files that were collected for offline analysis. You can also export data to Elasticsearch and Kibana (ELK stack)


Search Box
linux-conf-vm.c.velocidex-199308.internal
connected
0
mic
State
FlowId
Artifacts Collected
Creation Time
Last Active
Creator
F.BOAK95NB8HITQ
Linux.Syslog.SSHLogin
2020-01-08 02:55:18 UTC
2020-01-08 02:55:20 UTC
mic
Artifact Collection
Uploaded Files
Requests
Results
Log
Reports
Overview
Artifact Names
Linux.Syslog.SSHLogin
Flow ID
F.BOAK95NB8HITQ
Creator
mic
Start Time
2020-01-08 02:55:18 UTC
Last Active
2020-01-08 02:55:20 UTC
State
TERMINATED
Ops/Sec
Unlimited
Parameters
syslogAuthLogPath
/var/log/auth.log*
SSHGrok
%{SYSLOGTIMESTAMP:Timestamp} (?:%{SYSLOGFACILITY})?%{SYSLOGHOST:logsource} %{SYSLOGPROG}: %{DATA:event} %{DATA:method} for (invalid user)?%{DATA:user} from %{IPORHOST:ip} port %{NUMBER:port} ssh2(: %{GREEDYDATA:system.auth.ssh.signature})?
Results
Artifacts with Results
["Linux.Syslog.SSHLogin"]
Uploaded Bytes
0 / 0
Files uploaded
0
Download Results
Prepare Download
Available Downloads
F.BOAK95NB8HITQ.zip
14797 Bytes
2020-01-08 03:00:37.46471123 +0000 UTC

Velociraptor command line

Velociraptor offers many command line tools. e.g.:

List content of the zip files

```
velociraptor unzip --list F.BOAK95NB8HITQ.zip
```

Extract and filter collected data as JSON

```
velociraptor unzip --csv F.BOAK95NB8HITQ.zip
clients/C.77772029617b302a/artifacts/Linux.Syslog.SSHL
ogin/F.BOAK95NB8HITQ.csv
--where "AttemptedUser =~ 'mic'"
```

```
]mic@trek:~/Downloads$velociraptor unzip --list F.BOAK95NB8HITQ.zip
[
 {
  "Filename": "FlowDetails",
  "Size": 1119
 },
 {
  "Filename": "clients/C.77772029617b302a/collections/F.BOAK95NB8HITQ/logs",
  "Size": 360
 },
 {
  "Filename": "clients/C.77772029617b302a/artifacts/Linux.Syslog.SSHLogin/F.BOAK95NB8HITQ.csv",
  "Size": 92106
 }
]mic@trek:~/Downloads$ velociraptor unzip --csv F.BOAK95NB8HITQ.zip clients/C.77772029617b302a/artifacts/Li
nux.Syslog.SSHLogin/F.BOAK95NB8HITQ.csv --where "AttemptedUser =~ 'mic'"
[
 {
  "Time": "2020-01-07T00:52:10Z",
  "IP": "122.109.172.84",
  "Result": "Accepted",
  "Method": "publickey",
  "AttemptedUser": "mic",
  "FullPath": "/var/log/auth.log"
 },
 {
  "Time": "2020-01-07T00:52:29Z",
  "IP": "122.109.172.84",
  "Result": "Accepted",
  "Method": "publickey",
  "AttemptedUser": "mic",
  "FullPath": "/var/log/auth.log"
 }
mic@trek:~/Downloads$
```

ses

Hunting across the entire deployment

What is hunting?

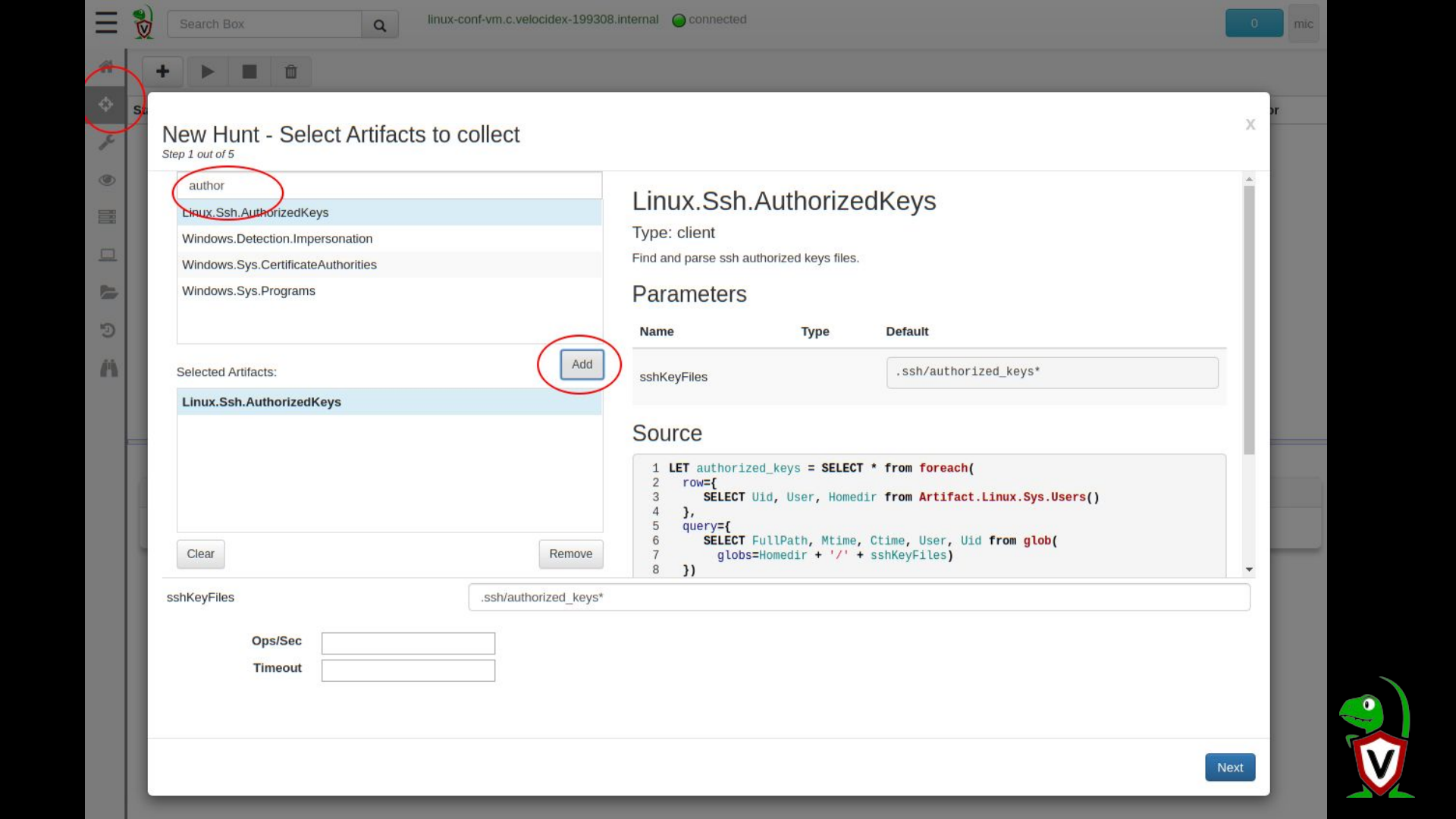
Any artifact that can be collected on a single computer, can be simultaneously hunted across the entire network.

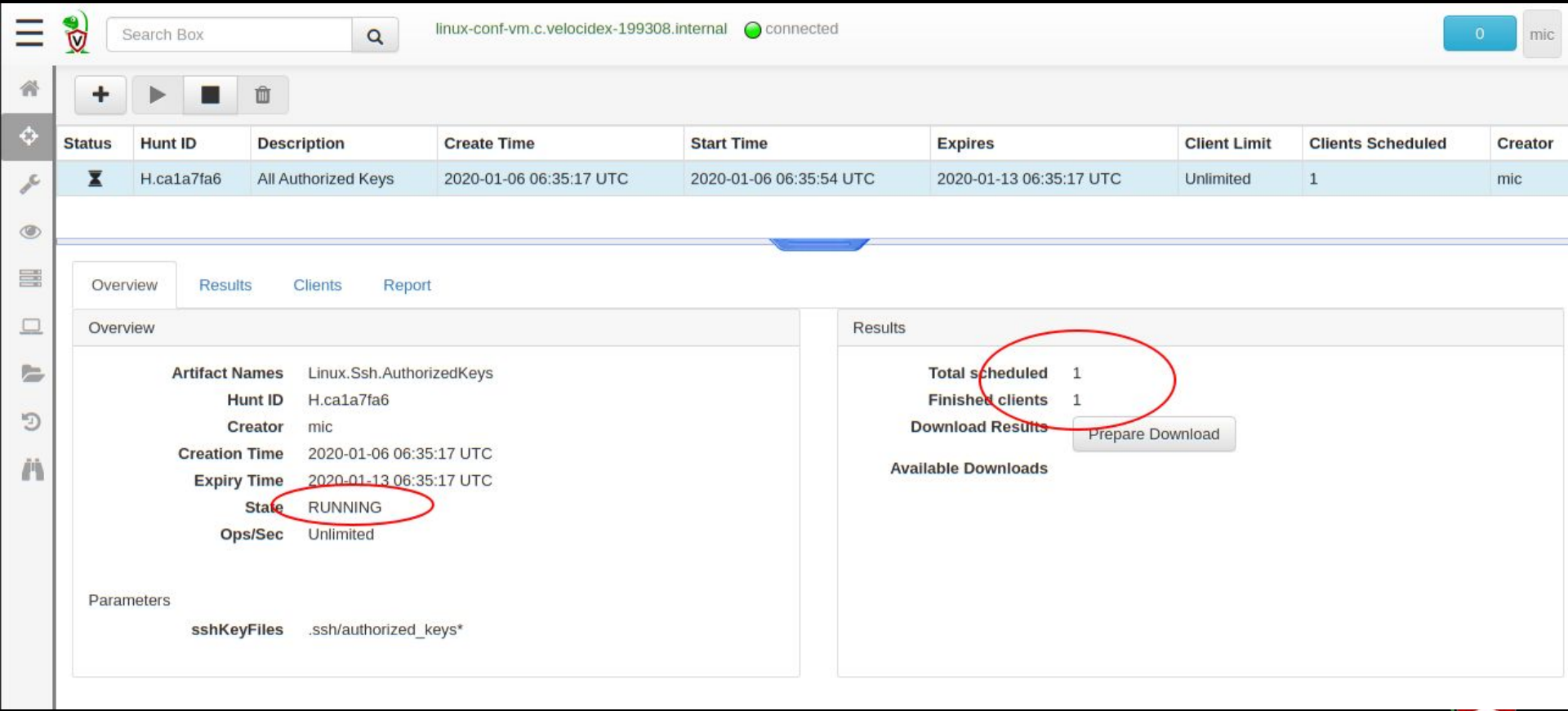
A hunt can cover a group of clients, or the whole network.

A hunt will continue running until it expires, or is stopped.

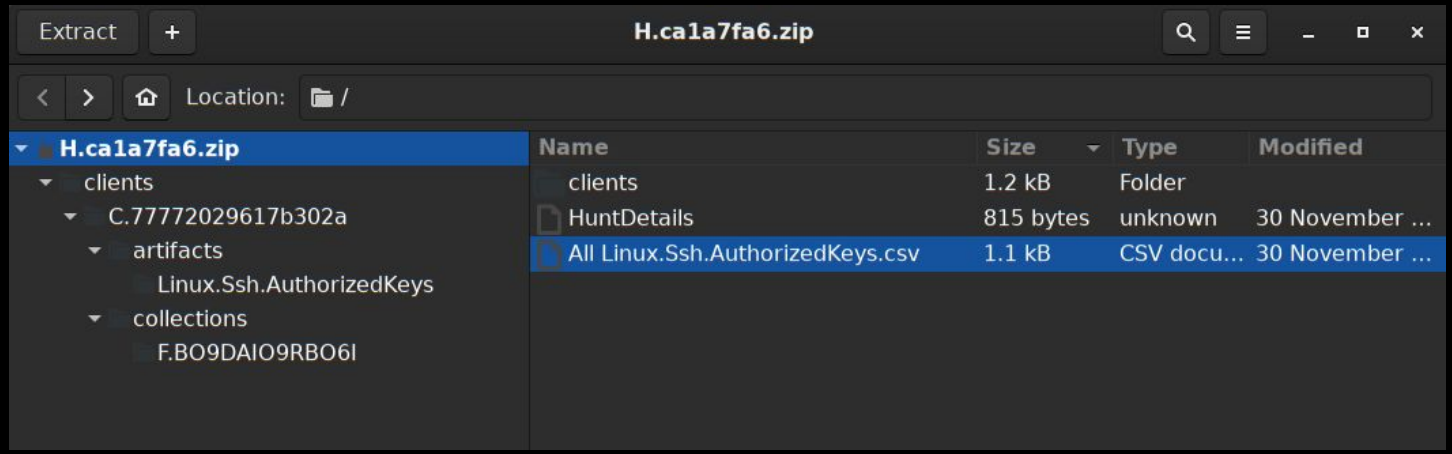
As new machines appear, they automatically join in the hunt.

Downloading the hunt results generates a ZIP file with all the uploaded files as well as a large CSV or JSON file with combined results from each client in the same file.


Search Box
linux-conf-vm.c.velocidex-199308.internal
connected
New Hunt - Select Artifacts to collect
Step 1 out of 5
author
Linux.Ssh.AuthorizedKeys
Windows.Detection.Impersonation
Windows.Sys.CertificateAuthorities
Windows.Sys.Programs
Add
Selected Artifacts:
Linux.Ssh.AuthorizedKeys
Clear
Remove
Linux.Ssh.AuthorizedKeys
Type: client
Find and parse ssh authorized keys files.
Parameters
Name
Type
Default
sshKeyFiles
.ssh/authorized_keys*
Source
LET authorized_keys = SELECT * from foreach(
row={
SELECT Uid, User, Homedir from Artifact.Linux.Sys.Users()
},
query={
SELECT FullPath, Mtime, Ctime, User, Uid from glob(
globs=Homedir + '/' + sshKeyFiles)
})
sshKeyFiles
.ssh/authorized_keys*
Ops/Sec
Timeout
Next



The ZIP file contains a single CSV file with combined output from all clients



What access does a Key have?

A single key can give access to multiple user accounts of many systems. Our hunt collects all the authorized keys on our endpoints in a large CSV file. We can search for the hosts that the key will allow access to.

```
]mic@trek:~/Downloads$ velociraptor unzip --csv H.ca1a7fa6.zip "All Linux.Ssh.AuthorizedKeys.csv"  --where "Key =~ 'Gu5BusVC5VWlPA7x86hdXU'"
[
 {
  "Uid": "1000",
  "User": "mic",
  "FullPath": "/home/mic/.ssh/authorized_keys",
  "Key": "AAAAB3NzaC1yc2EAAAADAQABAAABAQDELIxd5FPOcE4CAEAvUNSQVNxHobfOdtZHYkSRUYxcq6TZrigPmpQztF9MUU/mKT5x22MHnOZOceY2/lAn5ngMtYUEB6sHjSgFVHshKsrX
Hk8PsU/gWqjId8OIvMjRwwL9mJozpuyaTMwR5bWkqbJPJ4+V9mNLnGd9st+pOWFJFKVuX0NJYJ1BgYXgHRdY6oy/2maKtEI3CJeLSXz87K515tHBepVk6hSSRe8Mi0SV9EmcbH2kOaDwi74N65
d7s0d+I6QXo2KO6EOrBpKEtd8nRRqPbL56ICbGu5BusVC5VWlPA7x86hdXU+qfx+Pj",
  "Comment": "mic@trek\n",
  "Mtime": "2020-01-05T00:52:08Z",
  "FlowId": "F.BO9DAIO9RBO6I",
  "ClientId": "C.77772029617b302a",
  "Fqdn": "linux-conf-vm.c.velocidex-199308.internal"
 },
 {
  "Uid": "1001",
  "User": "scudette",
  "FullPath": "/home/scudette/.ssh/authorized_keys",
  "Key": "AAAAB3NzaC1yc2EAAAADAQABAAABAQDELIxd5FPOcE4CAEAvUNSQVNxHobfOdtZHYkSRUYxcq6TZrigPmpQztF9MUU/mKT5x22MHnOZOceY2/lAn5ngMtYUEB6sHjSgFVHshKsrX
Hk8PsU/gWqjId8OIvMjRwwL9mJozpuyaTMwR5bWkqbJPJ4+V9mNLnGd9st+pOWFJFKVuX0NJYJ1BgYXgHRdY6oy/2maKtEI3CJeLSXz87K515tHBepVk6hSSRe8Mi0SV9EmcbH2kOaDwi74N65
d7s0d+I6QXo2KO6EOrBpKEtd8nRRqPbL56ICbGu5BusVC5VWlPA7x86hdXU+qfx+Pj",
  "Comment": "mic@trek\n",
  "Mtime": "2020-01-06T05:42:15Z",
  "FlowId": "F.BO9DAIO9RBO6I",
  "ClientId": "C.77772029617b302a",
  "Fqdn": "linux-conf-vm.c.velocidex-199308.internal"
 }
```

Surgical collection of evidence

Trying to keep one step ahead

- Proactive DFIR work involves trying to keep ahead of current Tools Technique and Procedures (TTP)
- Reading a lot of blog posts
- Following the Mitre Att&ck framework

Ultimately we are looking for signals we can use to alert when an endpoint is compromised.

MATRICES
PRE-ATT&CK
Enterprise
Windows
macOS
Linux
Cloud
Mobile

Launch the ATT&CK™ Navigator

Linux Matrix

Below are the tactics and technique representing the MITRE ATT&CK Matrix™ for Enterprise. The Matrix contains information for the Linux platform.

Last Modified: 2019-10-09 18:48:31.906000

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Drive-by Compromise	Command-Line Interface	.bash_profile and .bashrc	Exploitation for Privilege Escalation	Binary Padding	Bash History	Account Discovery	Application Deployment Software	Audio Capture	Commonly Used Port	Automated Exfiltration	Account Access Removal
Exploit Public-Facing Application	Exploitation for Client Execution	Bootkit	Process Injection	Clear Command History	Brute Force	Browser Bookmark Discovery	Exploitation of Remote Services	Automated Collection	Communication Through Removable Media	Data Compressed	Data Destruction
Hardware Additions	Graphical User Interface	Browser Extensions	Setuid and Setgid	Compile After Delivery	Credential Dumping	File and Directory Discovery	Internal Spearphishing	Clipboard Data	Connection Proxy	Data Encrypted	Data Encrypted for Impact
Spearphishing Attachment	Local Job Scheduling	Create Account	Sudo	Connection Proxy	Credentials from Web Browsers	Network Service Scanning	Remote File Copy	Data from Information Repositories	Custom Command and Control Protocol	Data Transfer Size Limits	Defacement
Spearphishing Link	Scripting	Hidden Files and Directories	Sudo Caching	Disabling Security Tools	Credentials in Files	Network Sniffing	Remote Services	Data from Local System	Custom Cryptographic Protocol	Exfiltration Over Alternative Protocol	Disk Content Wipe
Spearphishing via Service	Source	Kernel Modules and Extensions	Valid Accounts	Execution Guardrails	Exploitation for Credential Access	Password Policy Discovery	SSH Hijacking	Data from Network Shared Drive	Data Encoding	Exfiltration Over Command and Control Channel	Disk Structure Wipe

ENTERPRISE

TECHNIQUES

Overview
Initial Access
Execution
Persistence
Privilege Escalation
Defense Evasion
Credential Access
Account Manipulation
Bash History
Brute Force
Cloud Instance Metadata API
Credential Dumping
Credentials from Web Browsers
Credentials in Files
Credentials in

Private Keys

Private cryptographic keys and certificates are used for authentication, encryption/decryption, and digital signatures. [1]

Adversaries may gather private keys from compromised systems for use in authenticating to Remote Services like SSH or for use in decrypting other collected files such as email. Common key and certificate file extensions include: .key, .pgp, .gpg, .ppk., .p12, .pem, .pfx, .cer, .p7b, .asc. Adversaries may also look in common key directories, such as `~/.ssh` for SSH keys on * nix-based systems or `C:\Users(username).ssh\` on Windows.

Private keys should require a password or passphrase for operation, so an adversary may also use Input Capture for keylogging or attempt to Brute Force the passphrase off-line.

Adversary tools have been discovered that search compromised systems for file extensions relating to cryptographic keys and certificates. [2] [3]

ID: T1145
Tactic: Credential Access
Platform: Linux, macOS, Windows
Permissions Required: User
Data Sources: File monitoring
Contributors: Itzik Kotler, SafeBreach
Version: 1.0
Created: 14 December 2017
Last Modified: 18 July 2019

Procedure Examples

Name	Description
Ebury	Ebury has intercepted unencrypted private keys as well as private key pass-phrases. [6]
Empire	Empire can use modules like `Invoke-SessionGopher` to extract private key and session information.[5]
jRAT	jRAT can steal keys for VPNs and cryptocurrency wallets.[7]
Machete	Machete has scanned and looked for cryptographic keys and certificate file extensions. [8]

Find all un-encrypted private keys

- SSH private key files reside in ~/.ssh/id_rsa or ~/.ssh/id_dsa
- Encrypted files have a line like:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,278B4B1765B49ECF24679ABF589A60CC
```


Linux.Ssh.PrivateKeys

Type: client

SSH Private keys can be either encryted or unencrypted. Unencrypted private keys are more risky because an attacker can use them without needing to unlock them with a password.

This artifact searches for private keys in the usual locations and also records if they are encrypted or not.

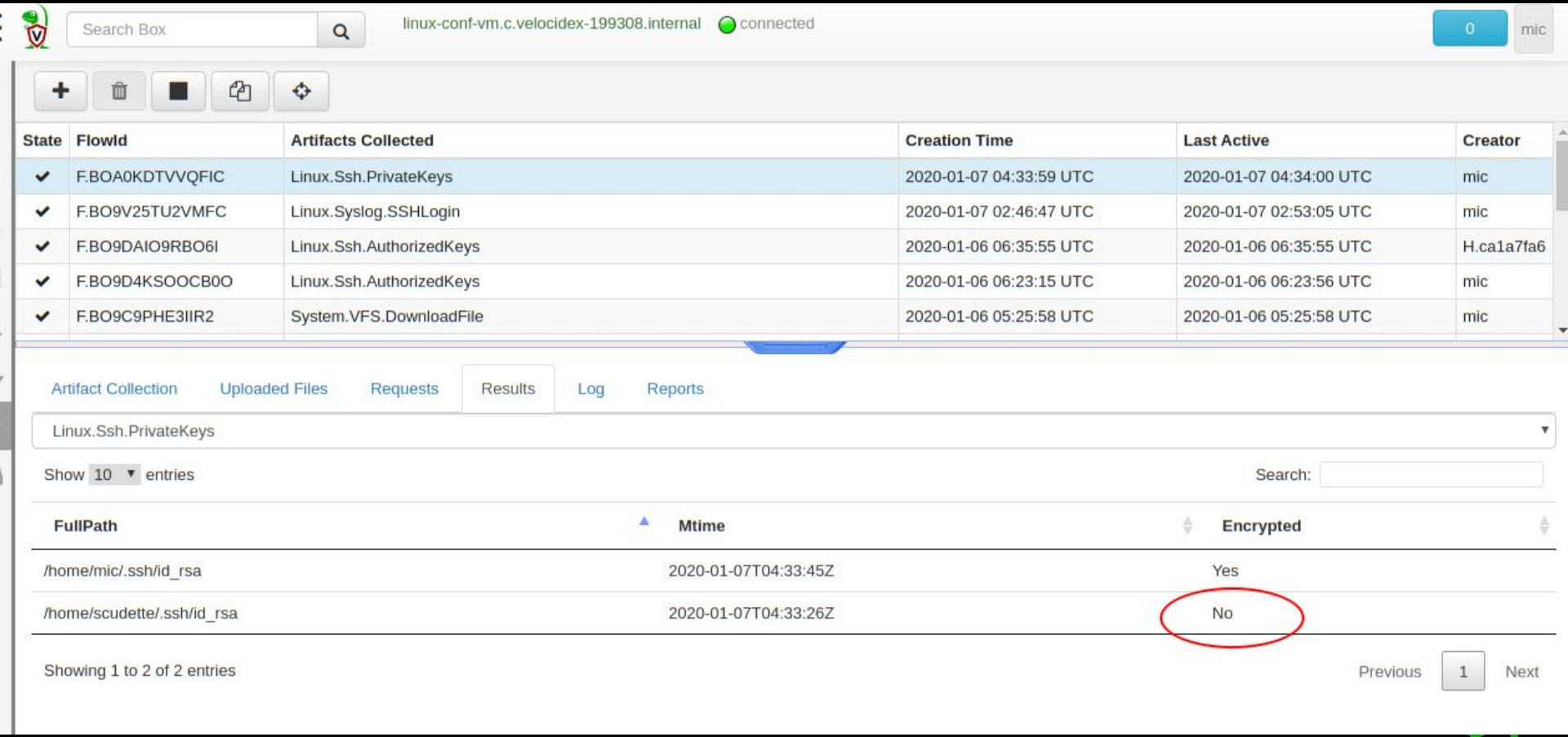
Parameters

Name	Type	Default
KeyGlobs		`/home/*/.ssh/id_{rsa,dsa}`

Source

```
SELECT FullPath,
       timestamp(epoch=Mtime.Sec) AS Mtime,
       if(condition={
            SELECT * from yara(rules="wide ascii:ENCRYPTED", files=FullPath)
          }, then="Yes", else="No") AS Encrypted
FROM glob(globs=KeyGlobs)

```



ENTERPRISE

ECHNIQUES
verview
itial Access
xecution
ersistence
.bash_profile and .bashrc
Accessibility Features
Account Manipulation
AppCert DLLs
AppInit DLLs
Application Shimming

Setuid and Setgid

When the setuid or setgid bits are set on Linux or macOS for an application, this means that the application will run with the privileges of the owning user or group respectively [1]. Normally an application is run in the current user's context, regardless of which user or group owns the application. There are instances where programs need to be executed in an elevated context to function properly, but the user running them doesn't need the elevated privileges. Instead of creating an entry in the sudoers file, which must be done by root, any user can specify the setuid or setgid flag to be set for their own applications. These bits are indicated with an "s" instead of an "x" when viewing a file's attributes via `ls -l`. The `chmod` program can set these bits with via bitmasking, `chmod 4777 [file]` or via shorthand naming, `chmod u+s [file]`.

An adversary can take advantage of this to either do a shell escape or exploit a vulnerability in an application with the setsuid or setgid bits to get code running in a different user's context. Additionally, adversaries can use this mechanism on their own malware to make sure they're able to execute in elevated contexts in the future [2].

ID: T1166

Tactic: Privilege Escalation, Persistence

Platform: Linux, macOS

Permissions Required: User

Effective Permissions: Administrator, root

Data Sources: File monitoring, Process monitoring, Process command-line parameters

Version: 1.0

Created: 14 December 2017

Last Modified: 24 June 2019

Create a SUID backdoor!

```
echo 'int main() { setresuid(0,0,0);
system("/bin/sh"); }' > privshell.c

gcc -o privshell privshell.c

rm privshell.c

chown root:root privshell

chmod u+s privshell
```

https://medium.com/@airman604/9-ways-to-backdoor-a-linux-box-f5f83bae5a3c

```
root@linux-conf-vm:/tmp# echo 'int main() { setresuid(0,0,0); system("/bin/sh"); }' > privshell.c
root@linux-conf-vm:/tmp# gcc -o privshell privshell.c
privshell.c: In function ‘main’:
privshell.c:1:14: warning: implicit declaration of function ‘setresuid’ [-Wimplicit-function-declaration]
 int main() { setresuid(0,0,0); system("/bin/sh"); }
              ^~~~~~~~~
privshell.c:1:32: warning: implicit declaration of function ‘system’ [-Wimplicit-function-declaration]
 int main() { setresuid(0,0,0); system("/bin/sh"); }
                                ^~~~~~
root@linux-conf-vm:/tmp# rm privshell.c
root@linux-conf-vm:/tmp# chown root:root privshell
root@linux-conf-vm:/tmp# chmod u+s privshell
```

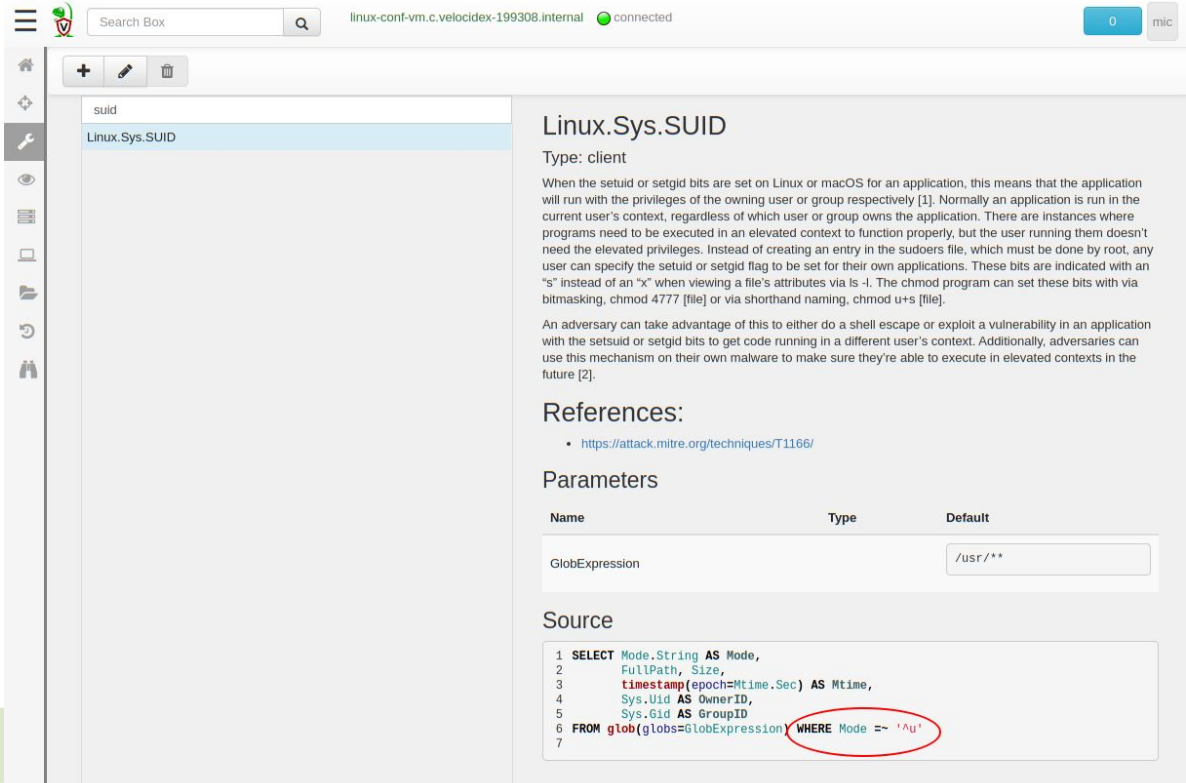
```
scudette@linux-conf-vm:~$ /tmp/privshell
# id
uid=0(root) gid=1002(scudette) groups=1002(scudette),4(
#
```

Hunt for SUID backdoors

Glob recursively for all files in a directory

Filter only those files with SUID bit enabled.

Could take a while on large system - throttle the endpoint if needed



New Artifact Collection - Select Artifacts to collect

Step 1 out of 2

suid

Linux.Sys.SUID

Add

Selected Artifacts:

Linux.Sys.SUID

Clear

Remove

execute in elevated contexts in the future [2].

References:

- https://attack.mitre.org/techniques/T1166/

Parameters

Name	Type	Default
GlobExpression		/usr/**

Source

```
SELECT Mode.String AS Mode,
       FullPath, Size,
       timestamp(epoch=Mtime.Sec) AS Mtime,
       Sys.Uid AS OwnerID,
       Sys.Gid AS GroupID
FROM glob(globs=GlobExpression) WHERE Mode =~ '^u'

```

GlobExpression: /**

Ops/Sec

Maximum Time: 600

Next

Search Box

linux-conf-vm.c.velocidex-199308.internal connected

0 mic

State	FlowId	Artifacts Collected	Creation Time	Last Active	Creator
✔	F.BOAJ2E48NTSQA	Linux.Sys.SUID	2020-01-08 01:32:40 UTC	2020-01-08 01:32:47 UTC	mic

Artifact Collection | Uploaded Files | Requests | Results | Log | Reports

Linux.Sys.SUID

Show 50 entries

Search:

Mode	FullPath	Size	Mtime	OwnerID	GroupID
urwxr-xr-x	/bin/mount	44304	2018-03-07T18:29:09Z	0	0
urwxr-xr-x	/bin/ping	61240	2016-11-10T06:23:32Z	0	0
urwxr-xr-x	/bin/su	40536	2017-05-17T11:59:59Z	0	0
urwxr-xr-x	/bin/umount	31720	2018-03-07T18:29:09Z	0	0
urwxr-xr-x	/usr/bin/chfn	50040	2017-05-17T11:59:59Z	0	0
urwxr-xr-x	/usr/bin/chsh	40504	2017-05-17T11:59:59Z	0	0
urwxr-xr-x	/usr/bin/gpasswd	75792	2017-05-17T11:59:59Z	0	0
urwxr-xr-x	/usr/bin/newgrp	40312	2017-05-17T11:59:59Z	0	0
urwxr-xr-x	/usr/bin/passwd	59680	2017-05-17T11:59:59Z	0	0
urwxr-xr-x	/usr/bin/sudo	140944	2019-10-12T14:20:21Z	0	0
urwxr-xr-x	/usr/lib/openssh/ssh-keysign	440728	2019-07-15T13:32:09Z	0	0
urwxr-xr-x	/tmp/privshell	8704	2020-01-08T01:23:45Z	0	0

Showing 1 to 12 of 12 entries

Previous 1 Next

Customizing artifacts

Customizing artifacts

Artifacts simply contain VQL statements.

It's easy to modify existing artifacts to your needs.

As you learn VQL, you can easily write your own.

Custom artifacts start with the **Custom** prefix.

You can use official or custom artifacts interchangeably.

You can also contribute your artifacts to the Velociraptor project

Customize Artifacts

The SUID artifact is great but we don't know if someone replaced one of the standard binaries

Lets calculate the hash of each SUID binary

This will show us outliers - some machines might have a different hash which is not known. Maybe VirusTotal know about it?

We need to edit the **Linux.Sys.SUID** artifact to add a hash column

suid

Linux.Sys.SUID

Linux.Sys.SUID

Type: client

When the setuid or setgid bits are set on Linux or macOS for an application, this means that the application will run with the privileges of the owning user or group respectively [1]. Normally an application is run in the current user's context, regardless of which user or group owns the application. There are instances where programs need to be executed in an elevated context to function properly, but the user running them doesn't need the elevated privileges. Instead of creating an entry in the sudoers file, which must be done by root, any user can specify the setuid or setgid flag to be set for their own applications. These bits are indicated with an "s" instead of an "x" when viewing a file's attributes via ls -l. The chmod program can set these bits with via bitmasking, chmod 4777 [file] or via shorthand naming, chmod u+s [file].

An adversary can take advantage of this to either do a shell escape or exploit a vulnerability in an application with the setsuid or setgid bits to get code running in a different user's context. Additionally, adversaries can use this mechanism on their own malware to make sure they're able to execute in elevated contexts in the future [2].

References:

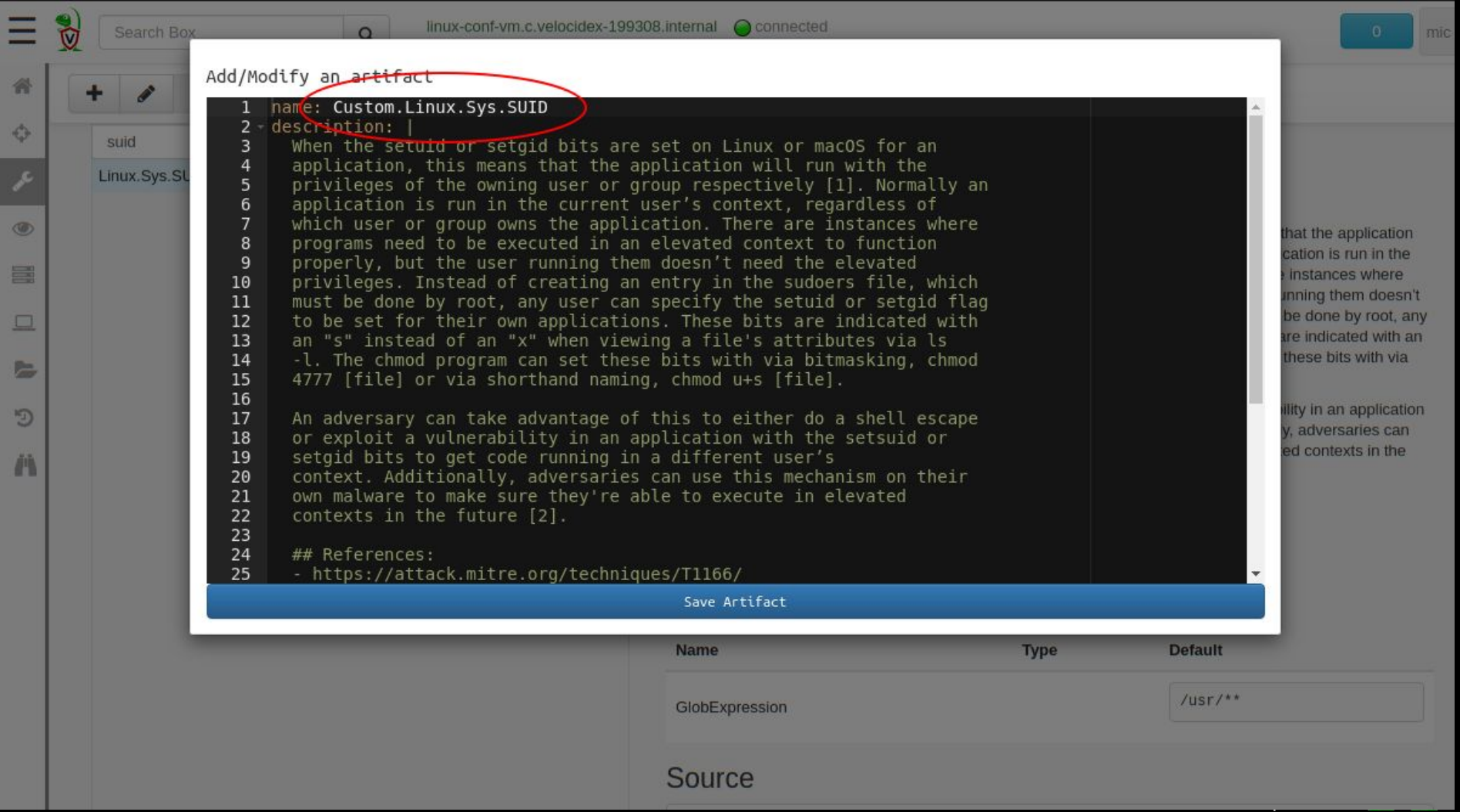
- https://attack.mitre.org/techniques/T1166/

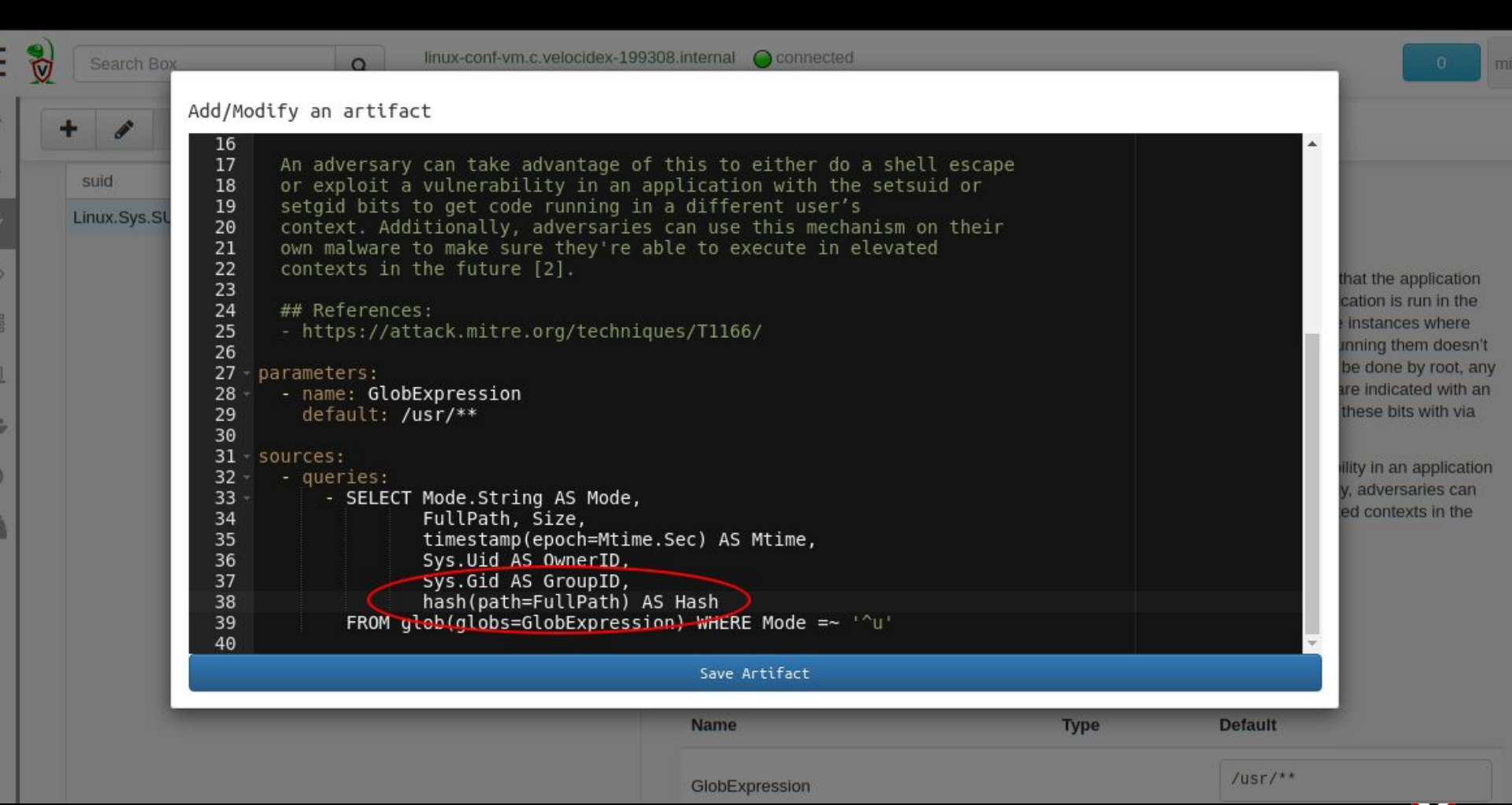
Parameters

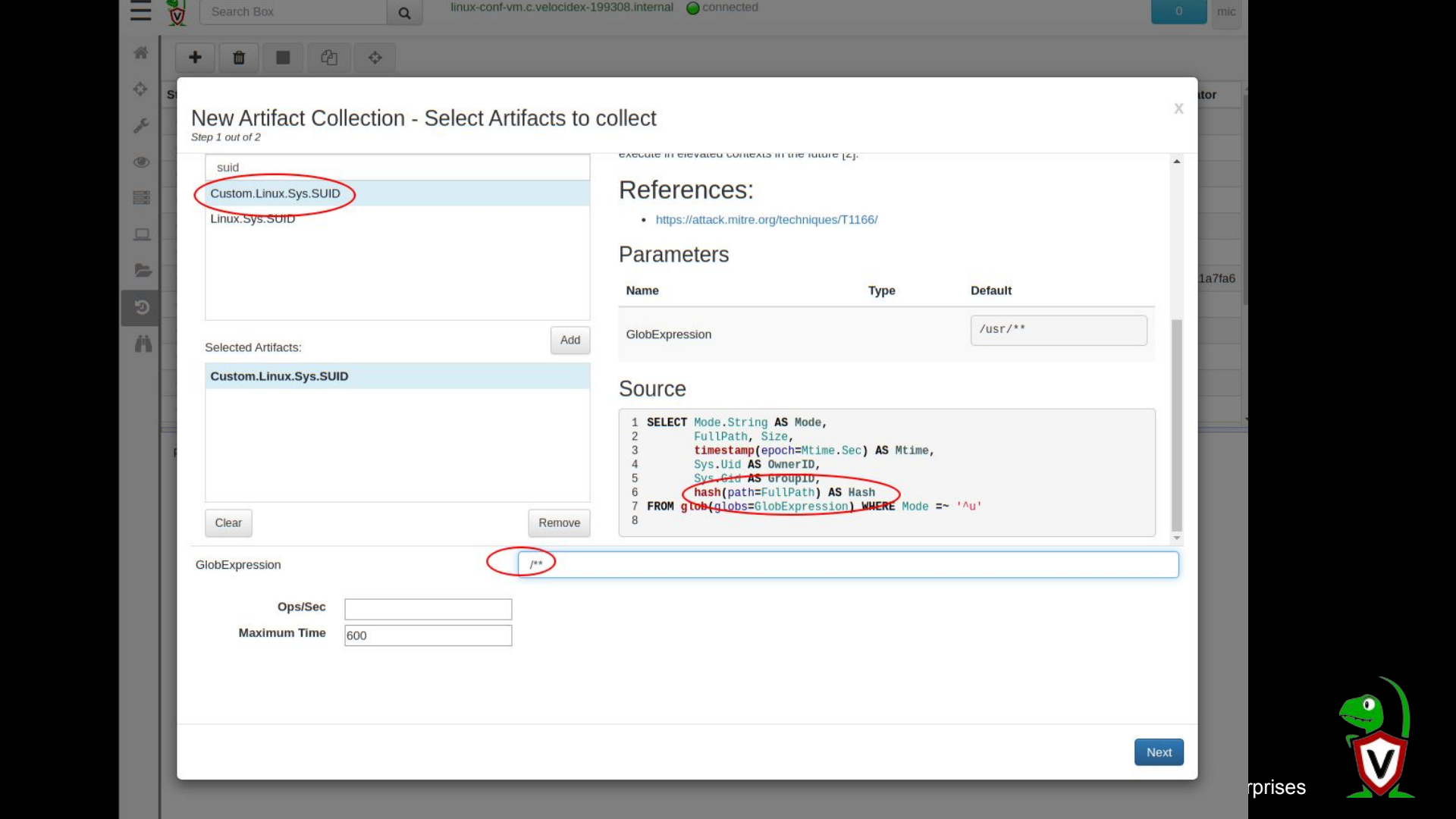
Name	Type	Default
GlobExpression		/usr/**

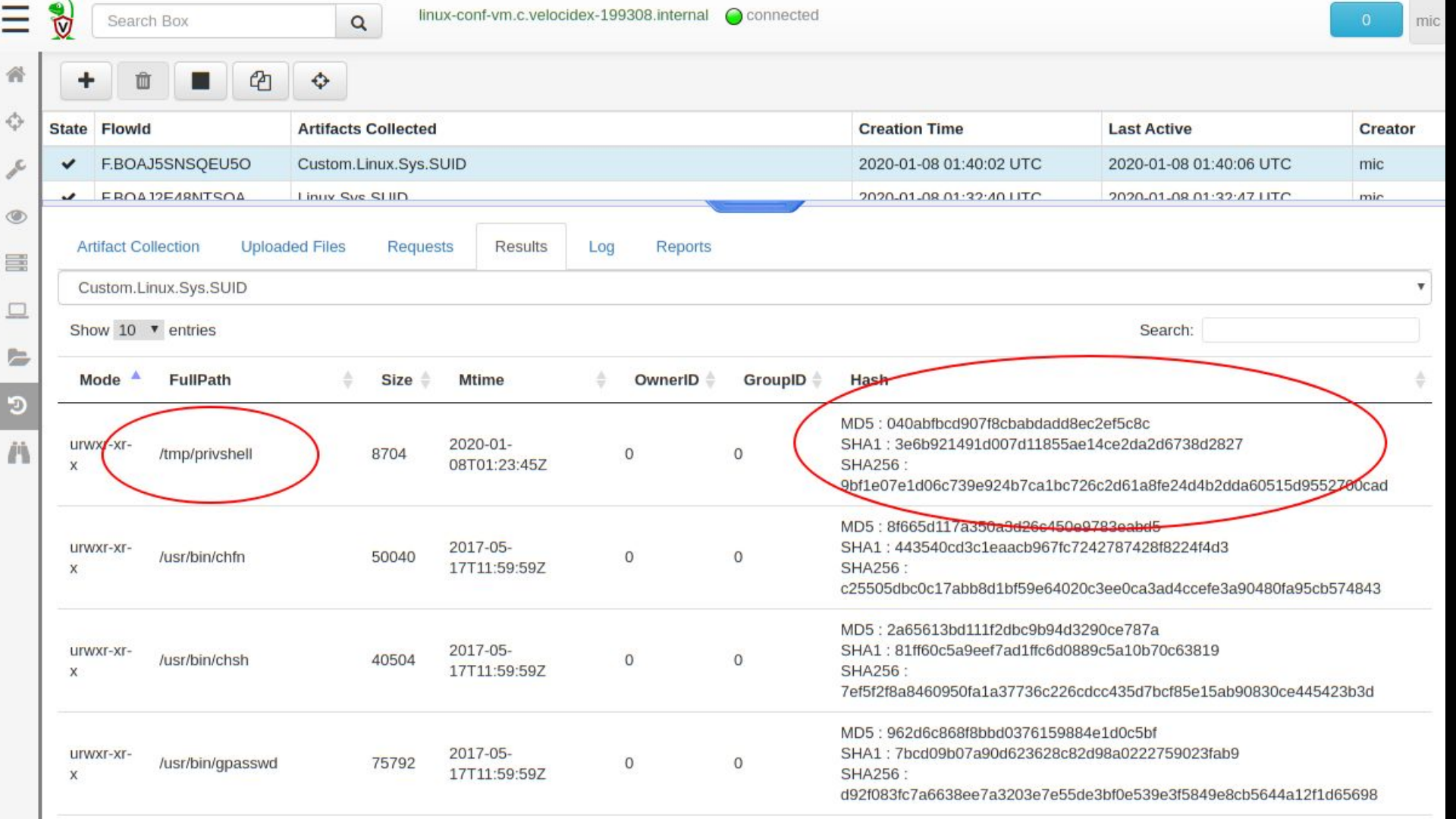
Source

```
1 SELECT Mode.String AS Mode,
2        FullPath, Size,
3        timestamp(epoch=Mtime.Sec) AS Mtime,
4        Sys.Uid AS OwnerID,
5        Sys.Gid AS GroupID
6 FROM glob(globs=GlobExpression) WHERE Mode =~ '^u'
7
```


Search Box
linux-conf-vm.c.velocidex-199308.internal
connected
0
mic
suid
Linux.Sys.SU
Add/Modify an artifact
name: Custom.Linux.Sys.SUID
description: |
When the setuid or setgid bits are set on Linux or macOS for an
application, this means that the application will run with the
privileges of the owning user or group respectively [1]. Normally an
application is run in the current user's context, regardless of
which user or group owns the application. There are instances where
programs need to be executed in an elevated context to function
properly, but the user running them doesn't need the elevated
privileges. Instead of creating an entry in the sudoers file, which
must be done by root, any user can specify the setuid or setgid flag
to be set for their own applications. These bits are indicated with
an "s" instead of an "x" when viewing a file's attributes via ls
-l. The chmod program can set these bits with via bitmasking, chmod
4777 [file] or via shorthand naming, chmod u+s [file].
An adversary can take advantage of this to either do a shell escape
or exploit a vulnerability in an application with the setsuid or
setgid bits to get code running in a different user's
context. Additionally, adversaries can use this mechanism on their
own malware to make sure they're able to execute in elevated
contexts in the future [2].
References:
- https://attack.mitre.org/techniques/T1166/
Save Artifact
Name
Type
Default
GlobExpression
/usr/**
Source




Search Box
linux-conf-vm.c.velocidex-199308.internal
connected
New Artifact Collection - Select Artifacts to collect
Step 1 out of 2
suid
Custom.Linux.Sys.SUID
Linux.Sys.SUID
Selected Artifacts:
Add
Custom.Linux.Sys.SUID
Clear
Remove
References:
https://attack.mitre.org/techniques/T1166/
Parameters
Name
Type
Default
GlobExpression
/usr/**
Source
SELECT Mode.String AS Mode,
FullPath, Size,
timestamp(epoch=Mtime.Sec) AS Mtime,
Sys.Uid AS OwnerID,
Sys.Gid AS GroupID,
hash(path=FullPath) AS Hash
FROM glob(globs=GlobExpression) WHERE Mode =~ '^u'
GlobExpression
/**
Ops/Sec
Maximum Time
600
Next


Search Box
linux-conf-vm.c.velocidex-199308.internal
connected
0
mic
State
FlowId
Artifacts Collected
Creation Time
Last Active
Creator
F.BOAJ5SNSQEU5O
Custom.Linux.Sys.SUID
2020-01-08 01:40:02 UTC
2020-01-08 01:40:06 UTC
mic
Artifact Collection
Uploaded Files
Requests
Results
Log
Reports
Custom.Linux.Sys.SUID
Show 10 entries
Search:
Mode
FullPath
Size
Mtime
OwnerID
GroupID
Hash
urwxr-xr-x
/tmp/privshell
8704
2020-01-08T01:23:45Z
0
0
MD5 : 040abfbcd907f8cbabdadd8ec2ef5c8c
SHA1 : 3e6b921491d007d11855ae14ce2da2d6738d2827
SHA256 :
9bf1e07e1d06c739e924b7ca1bc726c2d61a8fe24d4b2dda60515d9552700cad
urwxr-xr-x
/usr/bin/chfn
50040
2017-05-17T11:59:59Z
0
0
MD5 : 8f665d117a350a3d26c450e9783eabd5
SHA1 : 443540cd3c1eaacb967fc7242787428f8224f4d3
SHA256 :
c25505dbc0c17abb8d1bf59e64020c3ee0ca3ad4ccefe3a90480fa95cb574843
urwxr-xr-x
/usr/bin/chsh
40504
2017-05-17T11:59:59Z
0
0
MD5 : 2a65613bd111f2dbc9b94d3290ce787a
SHA1 : 81ff60c5a9eef7ad1ffc6d0889c5a10b70c63819
SHA256 :
7ef5f2f8a8460950fa1a37736c226cdcc435d7bcf85e15ab90830ce445423b3d
urwxr-xr-x
/usr/bin/gpasswd
75792
2017-05-17T11:59:59Z
0
0
MD5 : 962d6c868f8bbd0376159884e1d0c5bf
SHA1 : 7bcd09b07a90d623628c82d98a0222759023fab9
SHA256 :
d92f083fc7a6638ee7a3203e7e55de3bf0e539e3f5849e8cb5644a12f1d65698

Process injection

atomic-red-team/T10 ×

github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1055/T1055.md

Atomic Test #3 - Shared Library Injection via /etc/ld.so.preload

This test adds a shared library to the `ld.so.preload` list to execute and intercept API calls. This te threat actor Rocke during the exploitation of Linux web servers. This requires the `glibc` package.

Supported Platforms: Linux

Inputs

Name	Description	Type	Default Value
path_to_shared_library	Path to a shared library object	Path	../bin/T1055.so

Run it with `bash` ! Elevation Required (e.g. root or admin)

```
echo #{path_to_shared_library} > /etc/ld.so.preload
```

Let's test for this!
How can we detect it?

Sample code

```
/*
    Atomic Red Team Shared Object Library
    Uses code inspired by Zombie Ant Farm (https://github.com/dsnezhkov/zombieant)

    Compilation
    -------------
    gcc -shared -fPIC -o T1055.so T1055.c
*/

#include <stdio.h>

static void init(int argc, char **argv, char **envp) {
    printf("Loaded Atomic Red Team Library successfully!\n");
}

static void fini(void) {
    printf("Unloading Atomic Red Team preload...\n");
}


__attribute__((section(".init_array"), used)) static typeof(init) *init_p = init;
__attribute__((section(".fini_array"), used)) static typeof(fini) *fini_p = fini;
```

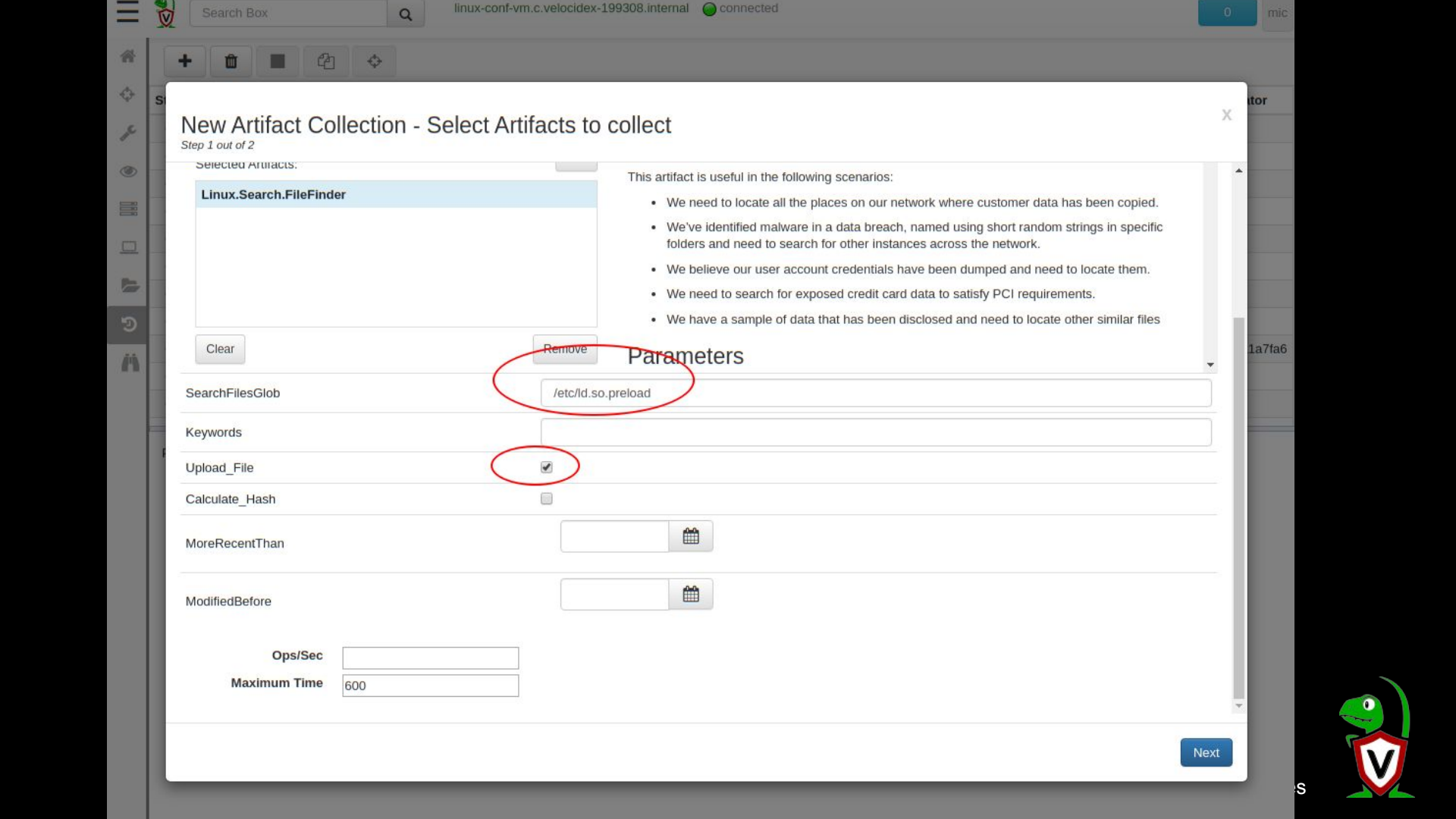
Copy/Paste this code and compile it into a shared object

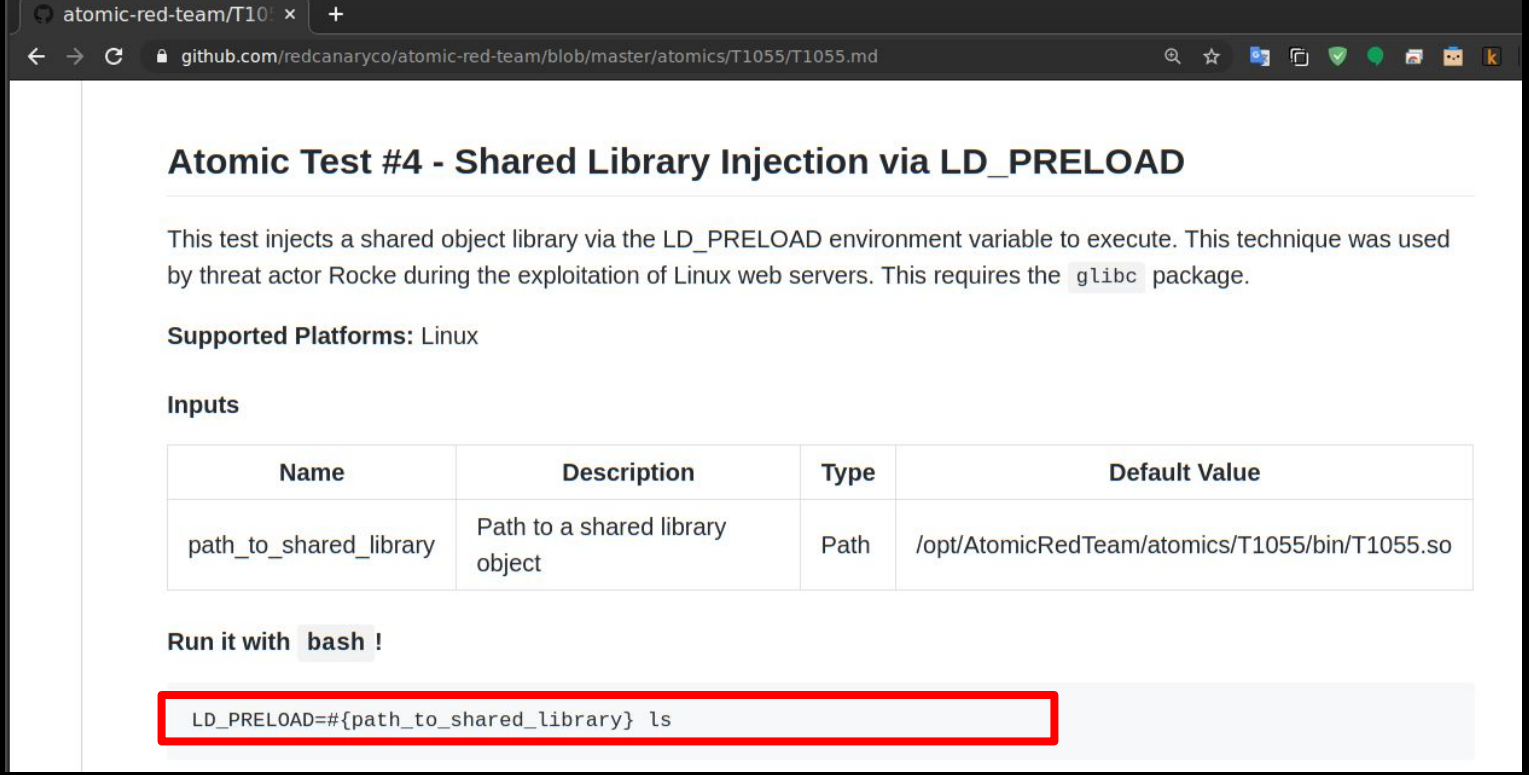
Detect global SO injections

How can we detect such process injection?

```
mic@DevBox:~/projects/atomic-red-team/atomics/T1055/src/Linux$ ls -l
Loaded Atomic Red Team Library successfully!
total 20
-rw-rw-r-- 1 mic mic   602 Jan  7 15:37 T1055.c
-rwxrwxr-x 1 mic mic 16064 Jan  7 15:38 T1055.so
mic@DevBox:~/projects/atomic-red-team/atomics/T1055/src/Linux$
```

The **Linux.Search.FileFinder** artifact can be used to search for /etc/ld.so.preload - it should not normally be there

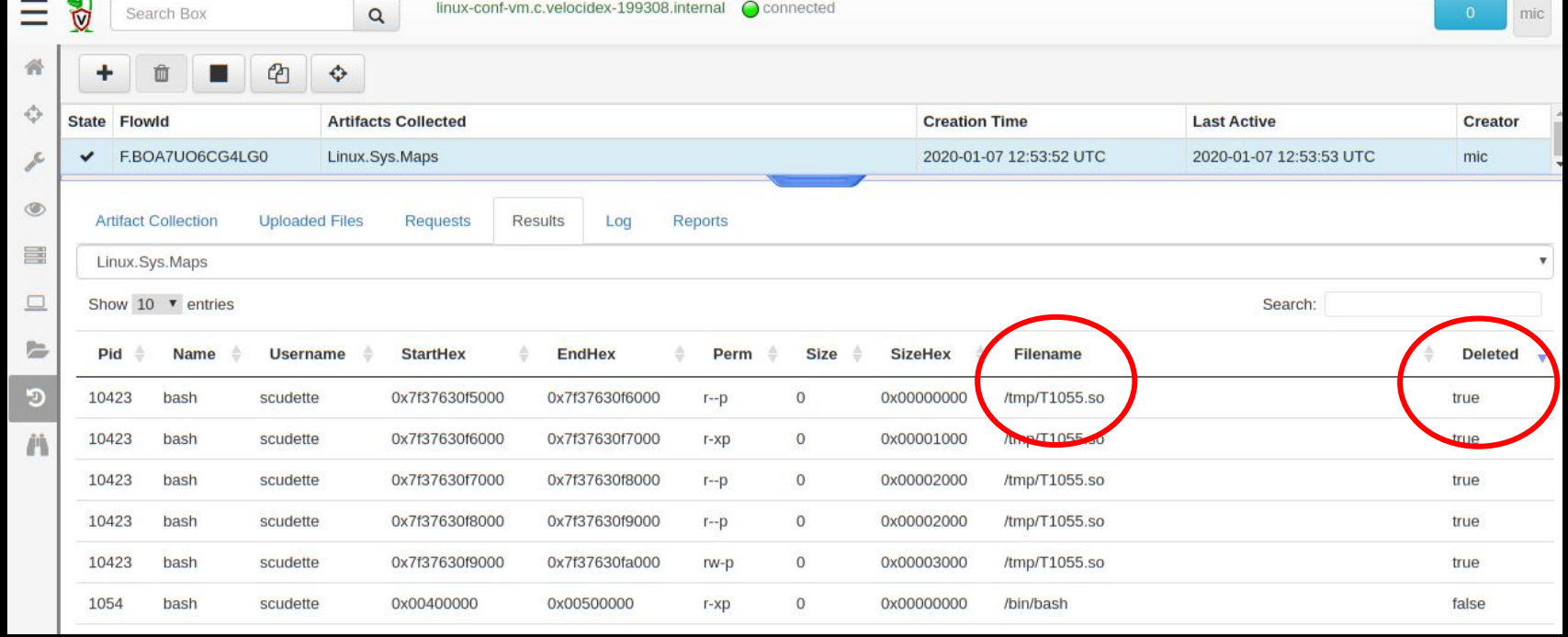

Search Box
linux-conf-vm.c.velocidex-199308.internal
connected
New Artifact Collection - Select Artifacts to collect
Step 1 out of 2
Selected Artifacts:
Linux.Search.FileFinder
Clear
Remove
This artifact is useful in the following scenarios:
We need to locate all the places on our network where customer data has been copied.
We've identified malware in a data breach, named using short random strings in specific folders and need to search for other instances across the network.
We believe our user account credentials have been dumped and need to locate them.
We need to search for exposed credit card data to satisfy PCI requirements.
We have a sample of data that has been disclosed and need to locate other similar files
Parameters
SearchFilesGlob
/etc/ld.so.preload
Keywords
Upload_File
Calculate_Hash
MoreRecentThan
ModifiedBefore
Ops/Sec
Maximum Time
600
Next



Let's launch a long lived process

LD_PRELOAD=/tmp/T1055.so bash

Now delete the shared object to make detection even harder!



We see that bash is linking a suspicious binary from /tmp which is also deleted!

Note: A binary linking a deleted library is not uncommon - it means the library was upgraded without the binary restarting. It also means the binary is still vulnerable if the library was updated to fix a security issue so it might be interesting to hunt for.

Inspecting with the GUI

The GUI is limited to show only 500 rows.

For larger result sets prepare and download the zip file containing the collected CSV files and parse them offline.

Scenario: Chrome extensions

Chrome extensions can be very dangerous.

They could access all website data including cookies and login creds.

They can create XSS opportunities for complete compromise.

Exfil is difficult to spot, since all communications occur over SSL.

Many Chrome extensions have been found to be malicious or vulnerable.

New Artifact Collection - Select Artifacts to collect

Step 1 out of 2

chrome

- Linux.Applications.Chrome.Extensions
- Linux.Applications.Chrome.Extensions.Upload
- Windows.Applications.Chrome.Cookies
- Windows.Applications.Chrome.Extensions
- Windows.Applications.Chrome.History

Add

Selected Artifacts:

Linux.Applications.Chrome.Extensions

Clear | Remove

Type: client

Fetch Chrome extensions.

Chrome extensions are installed into the user's home directory. We search for manifest.json files in a known path within each system user's home directory. We then parse the manifest file as JSON.

Many extensions use locale packs to resolve strings like name and description. In this case we detect the default locale and load those locale files. We then resolve the extension's name and description from there.

Parameters

Name	Type	Default
extensionGlobs		/.config/google-chrome/*/Extensions/*/*/manifest.json

Source

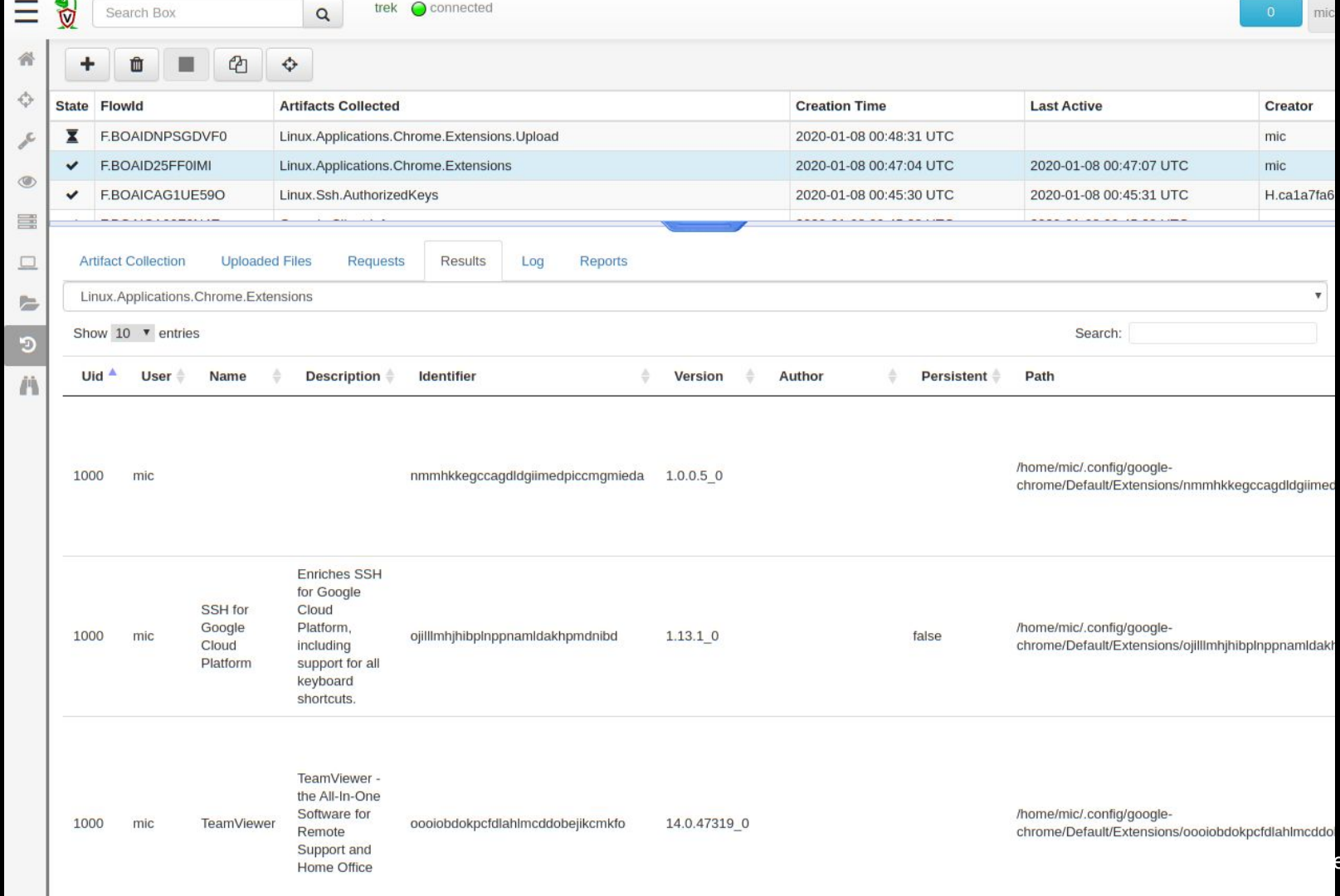
```
1 /* For each user on the system, search for extension manifests
2    in their home directory. */
3 LET extension_manifests = SELECT * from foreach(
4   row={
5     SELECT Uid, User, Homedir from Artifact.Linux.Sys.Users()
6   },
```

extensionGlobs: /.config/google-chrome/*/Extensions/*/*/manifest.json

Ops/Sec:

Maximum Time: 600

Next

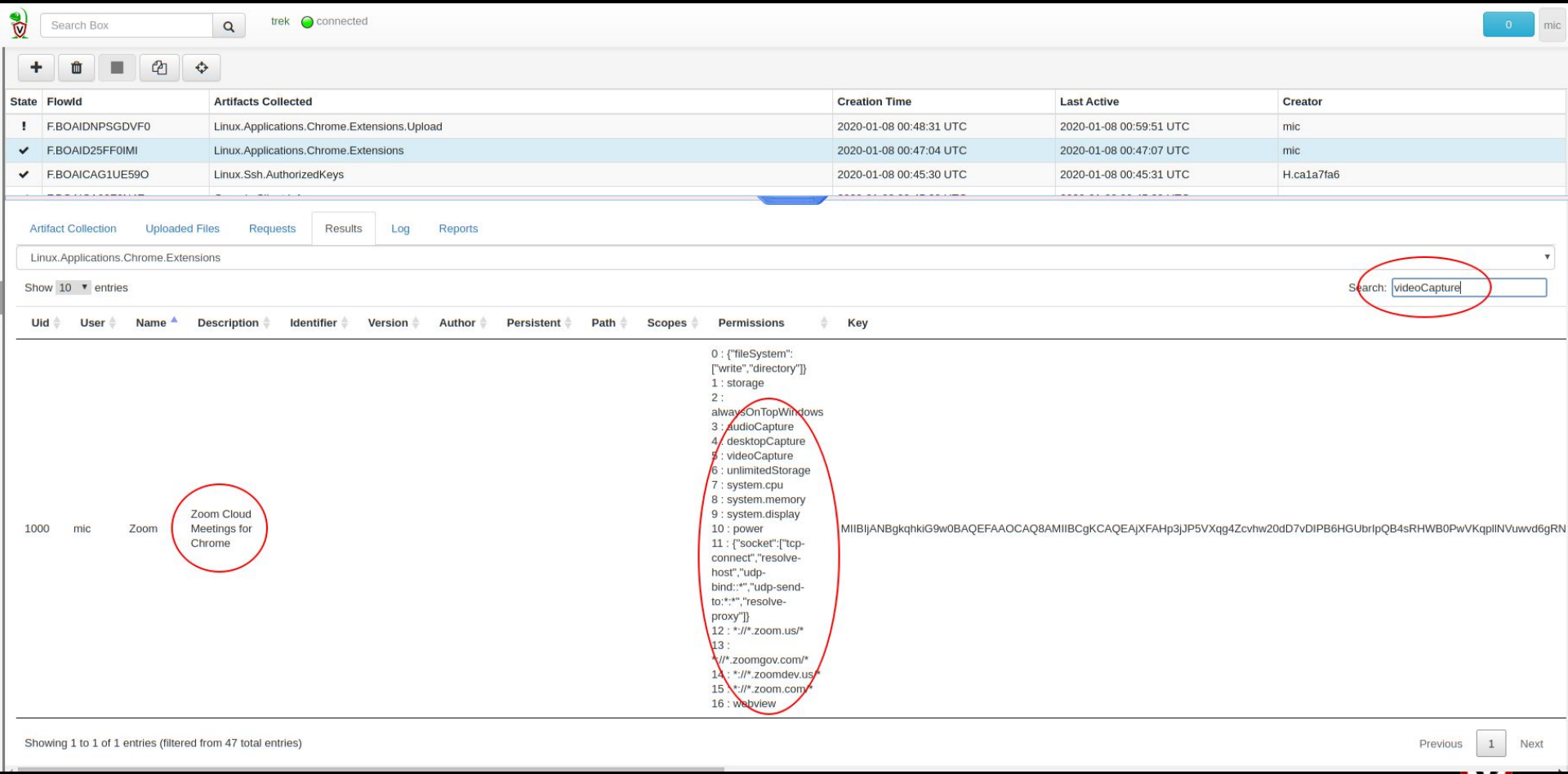

Search Box
trek
connected
0
mic
State
FlowId
Artifacts Collected
Creation Time
Last Active
Creator
F.BOAIDNPSGDVF0
Linux.Applications.Chrome.Extensions.Upload
2020-01-08 00:48:31 UTC
mic
F.BOAID25FF0IMI
Linux.Applications.Chrome.Extensions
2020-01-08 00:47:04 UTC
2020-01-08 00:47:07 UTC
mic
F.BOAICAG1UE59O
Linux.Ssh.AuthorizedKeys
2020-01-08 00:45:30 UTC
2020-01-08 00:45:31 UTC
H.ca1a7fa6
Artifact Collection
Uploaded Files
Requests
Results
Log
Reports
Linux.Applications.Chrome.Extensions
Show 10 entries
Search:
Uid
User
Name
Description
Identifier
Version
Author
Persistent
Path
1000
mic
nmmhkkegccagdldgiimedpiccmgmieda
1.0.0.5_0
/home/mic/.config/google-chrome/Default/Extensions/nmmhkkegccagdldgiimed
1000
mic
SSH for Google Cloud Platform
Enriches SSH for Google Cloud Platform, including support for all keyboard shortcuts.
ojilllmhjhibplnppnamldakhpmdnibd
1.13.1_0
false
/home/mic/.config/google-chrome/Default/Extensions/ojilllmhjhibplnppnamldakh
1000
mic
TeamViewer
TeamViewer - the All-In-One Software for Remote Support and Home Office
oooiobdokpcfdlahlmcddobejikcmkfo
14.0.47319_0
/home/mic/.config/google-chrome/Default/Extensions/oooiobdokpcfdlahlmcddo

Suspicious extensions?

Chrome extensions ask for permissions - some are dangerous!

- ❏ videoCapture allows the extension to take videos!
- ❏ socket allows extensions to make outbound connections!

It is worthwhile auditing chrome extensions in your organization with powerful permissions!



Suspicious chrome extension!

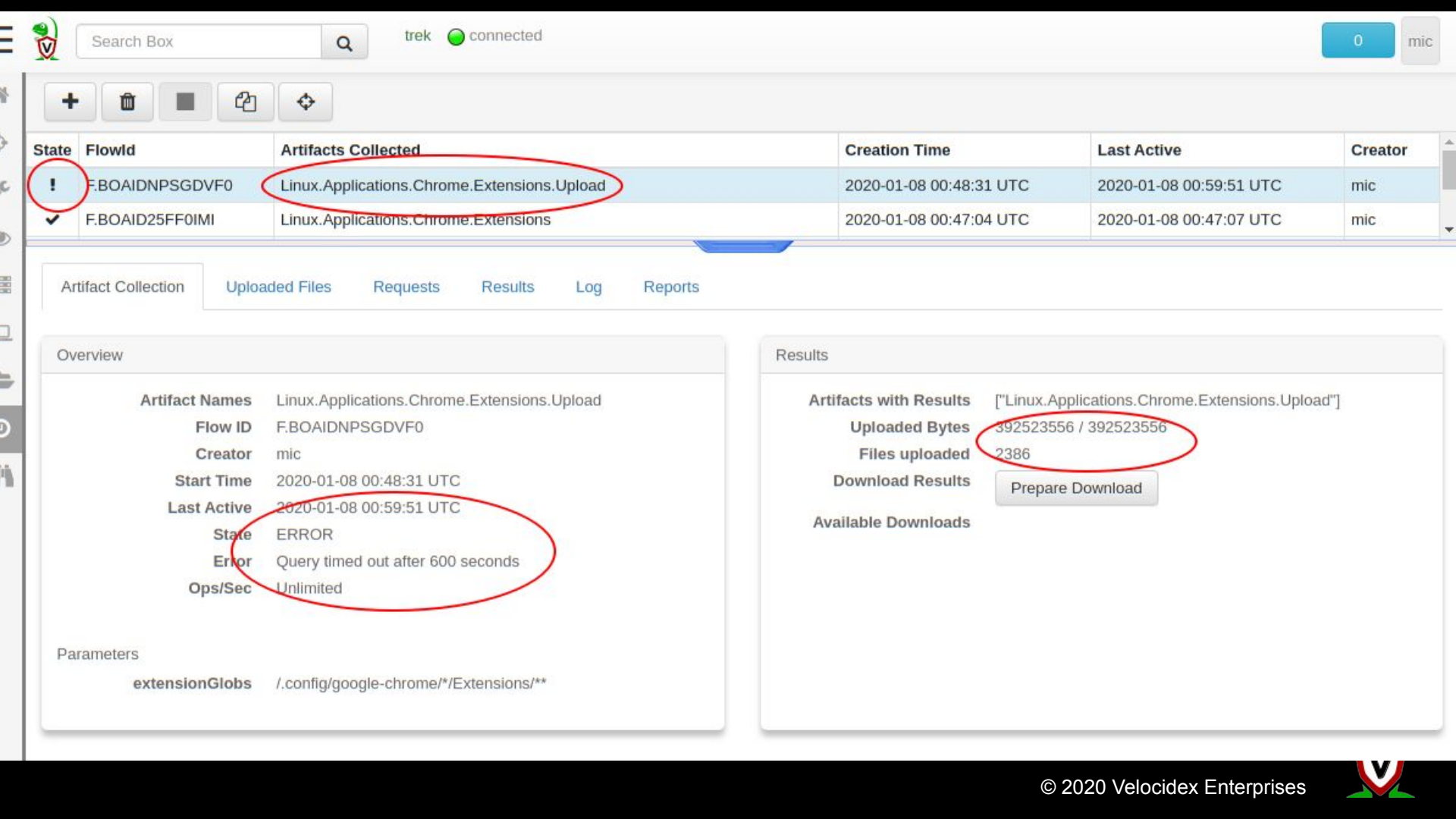
So you found a suspicious looking chrome extension?

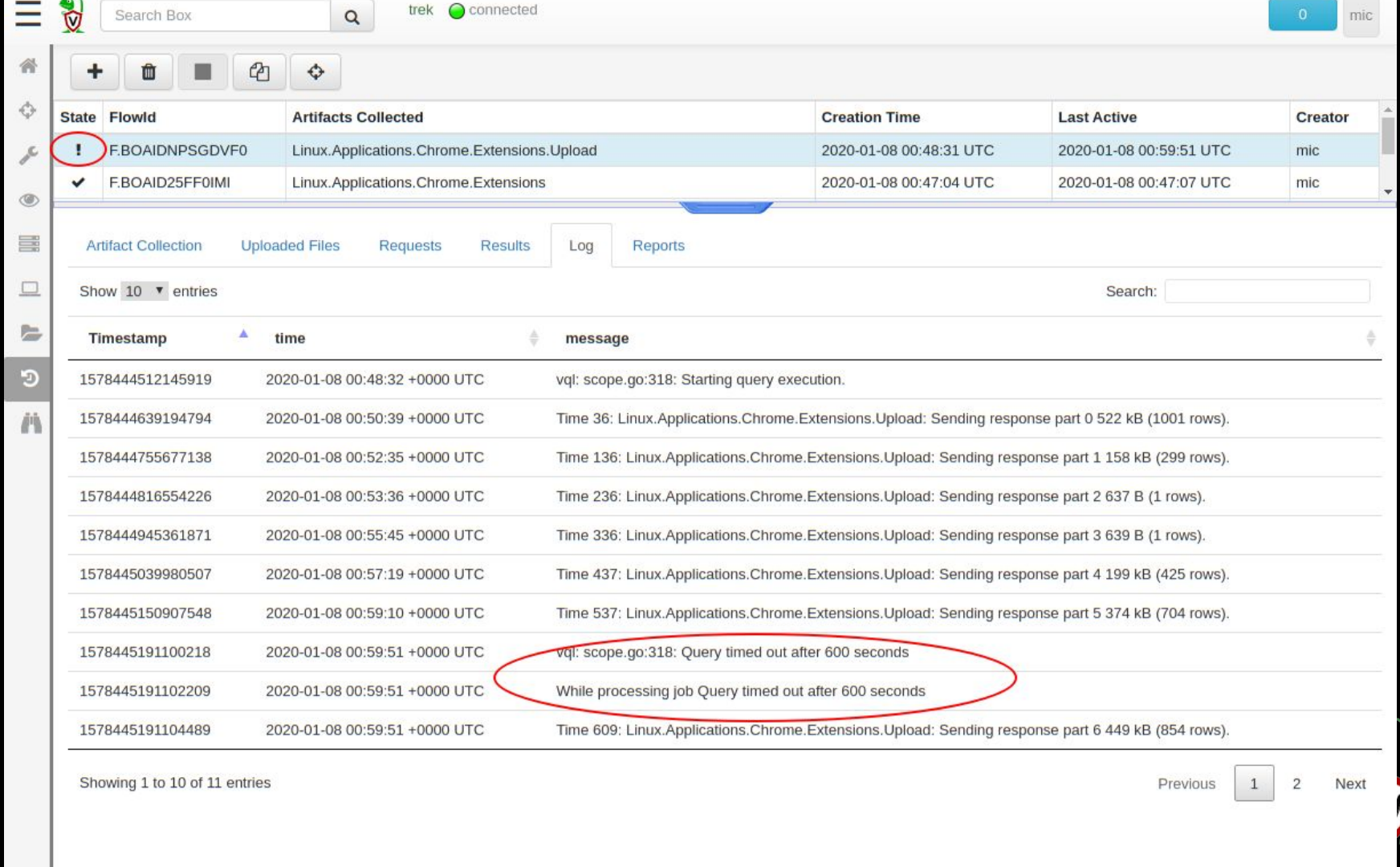
Upload the extension to the server for closer inspection, using the artifact:

Linux.Applications.Chrome.Extensions.Upload

Warning! This artifact gets all the files in all the extensions this could be huge!

On my system it yields over 400Mb!




Search Box
trek
connected
0
mic
State
FlowId
Artifacts Collected
Creation Time
Last Active
Creator
!
F.BOAIDNPSGDVF0
Linux.Applications.Chrome.Extensions.Upload
2020-01-08 00:48:31 UTC
2020-01-08 00:59:51 UTC
mic
✔
F.BOAID25FF0IMI
Linux.Applications.Chrome.Extensions
2020-01-08 00:47:04 UTC
2020-01-08 00:47:07 UTC
mic
Artifact Collection
Uploaded Files
Requests
Results
Log
Reports
Show 10 entries
Search:
Timestamp
time
message
1578444512145919
2020-01-08 00:48:32 +0000 UTC
vql: scope.go:318: Starting query execution.
1578444639194794
2020-01-08 00:50:39 +0000 UTC
Time 36: Linux.Applications.Chrome.Extensions.Upload: Sending response part 0 522 kB (1001 rows).
1578444755677138
2020-01-08 00:52:35 +0000 UTC
Time 136: Linux.Applications.Chrome.Extensions.Upload: Sending response part 1 158 kB (299 rows).
1578444816554226
2020-01-08 00:53:36 +0000 UTC
Time 236: Linux.Applications.Chrome.Extensions.Upload: Sending response part 2 637 B (1 rows).
1578444945361871
2020-01-08 00:55:45 +0000 UTC
Time 336: Linux.Applications.Chrome.Extensions.Upload: Sending response part 3 639 B (1 rows).
1578445039980507
2020-01-08 00:57:19 +0000 UTC
Time 437: Linux.Applications.Chrome.Extensions.Upload: Sending response part 4 199 kB (425 rows).
1578445150907548
2020-01-08 00:59:10 +0000 UTC
Time 537: Linux.Applications.Chrome.Extensions.Upload: Sending response part 5 374 kB (704 rows).
1578445191100218
2020-01-08 00:59:51 +0000 UTC
vql: scope.go:318: Query timed out after 600 seconds
1578445191102209
2020-01-08 00:59:51 +0000 UTC
While processing job Query timed out after 600 seconds
1578445191104489
2020-01-08 00:59:51 +0000 UTC
Time 609: Linux.Applications.Chrome.Extensions.Upload: Sending response part 6 449 kB (854 rows).
Showing 1 to 10 of 11 entries
Previous
1
2
Next

Velociraptor resource management

- Velociraptor is careful about the resource usage on the endpoint
  - Queries are timed out in 10 minutes by default
  - Queries can be rate limited on the endpoint - a notional ops/sec can be specified.
  - An op is a row or 1mb of scanned data

New Artifact Collection - Select Artifacts to collect

Step 1 out of 2

chrome

- Linux.Applications.Chrome.Extensions
- Linux.Applications.Chrome.Extensions.Upload
- Windows.Applications.Chrome.Cookies
- Windows.Applications.Chrome.Extensions
- Windows.Applications.Chrome.History

Add

Selected Artifacts:

Linux.Applications.Chrome.Extensions.Upload

Clear

Remove

Linux.Applications.Chrome.Extensions.Upload

Type: client

Upload all users chrome extension.

We dont bother actually parsing anything here, we just grab all the extension files in user's home directory.

Parameters

Name	Type	Default
extensionGlobs		/.config/google-chrome/*/Extensions/**

Source

```
1 /* For each user on the system, search for extension files
2    in their home directory and upload them. */
3 SELECT * from foreach(
4   row={
5     SELECT Uid, User, Homedir from Artifact.Linux.Sys.Users()
```

extensionGlobs /.config/google-chrome/*/Extensions/**

Ops/Sec 30

Maximum Time 1200

Event artifacts and endpoint monitoring

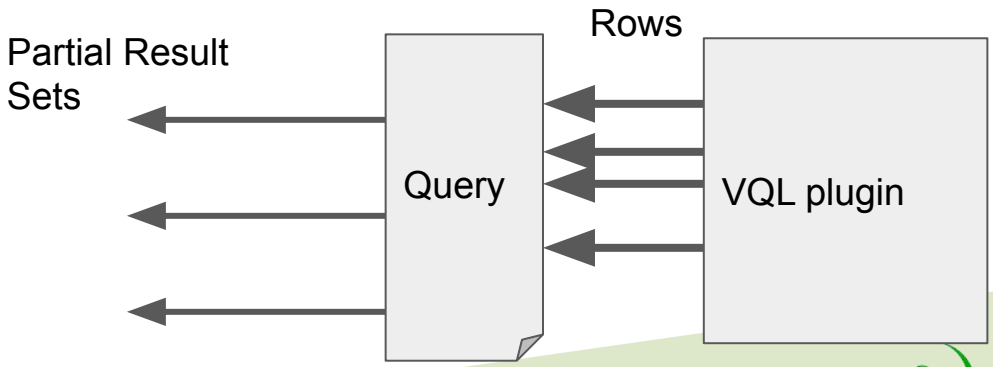
What are event artifacts?

Event artifacts are never-ending VQL queries that watch for events on clients and stream those events to the server.

Example:

Generic.Client.Stats

Samples CPU and Memory footprint every 10 seconds and streams to the server



desktop

DESKTOP-6CBJ8MJ connected

0 mic

Generic.Client.Stats

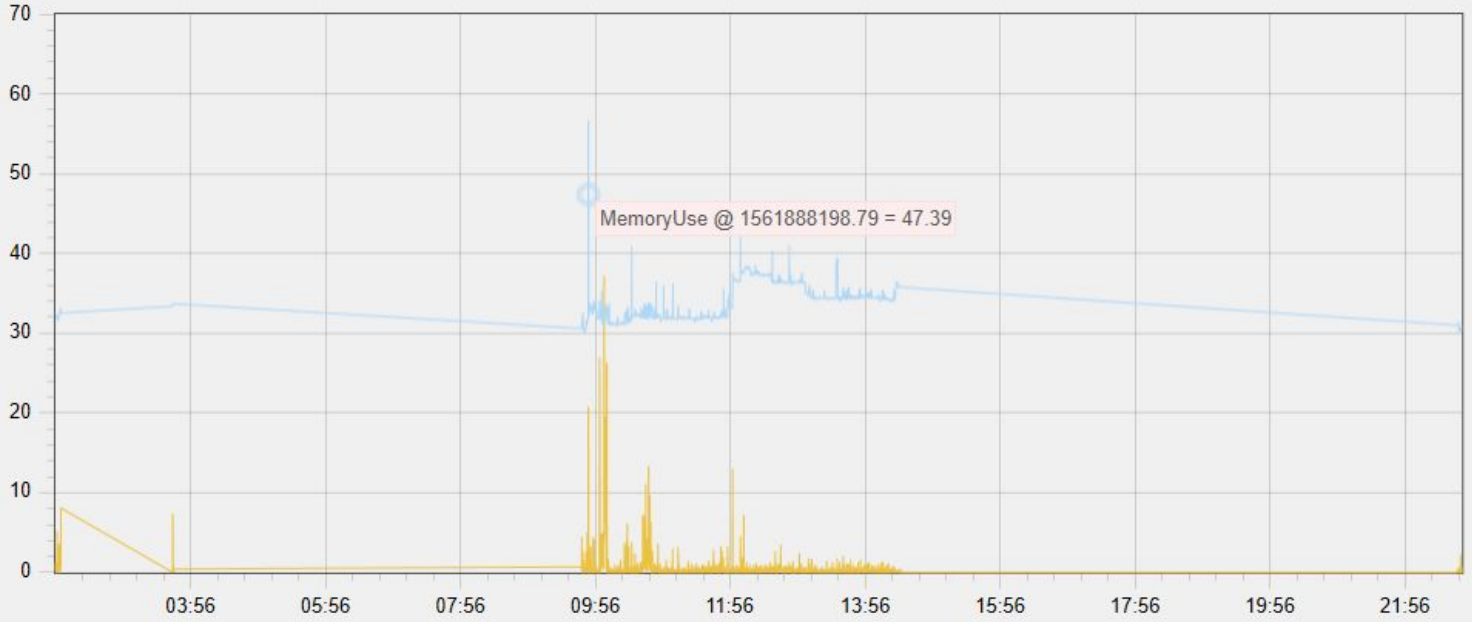
2019-06-30

Client Footprint for DESKTOP-6CBJ8MJ

The client has a client ID of C.e57080a99511ee58. Clients report the Velociraptor process footprint to the server every 10 seconds. The data includes the total CPU utilization, and the resident memory size used by the client.

The following graph shows the total utilization. Memory utilization is meausred in `Mb` while CPU Utilization is measured by `Percent of one core`.

We would expect the client to use around 1-5% of one core when idle, but if a heavy hunt is running this might climb substantially.



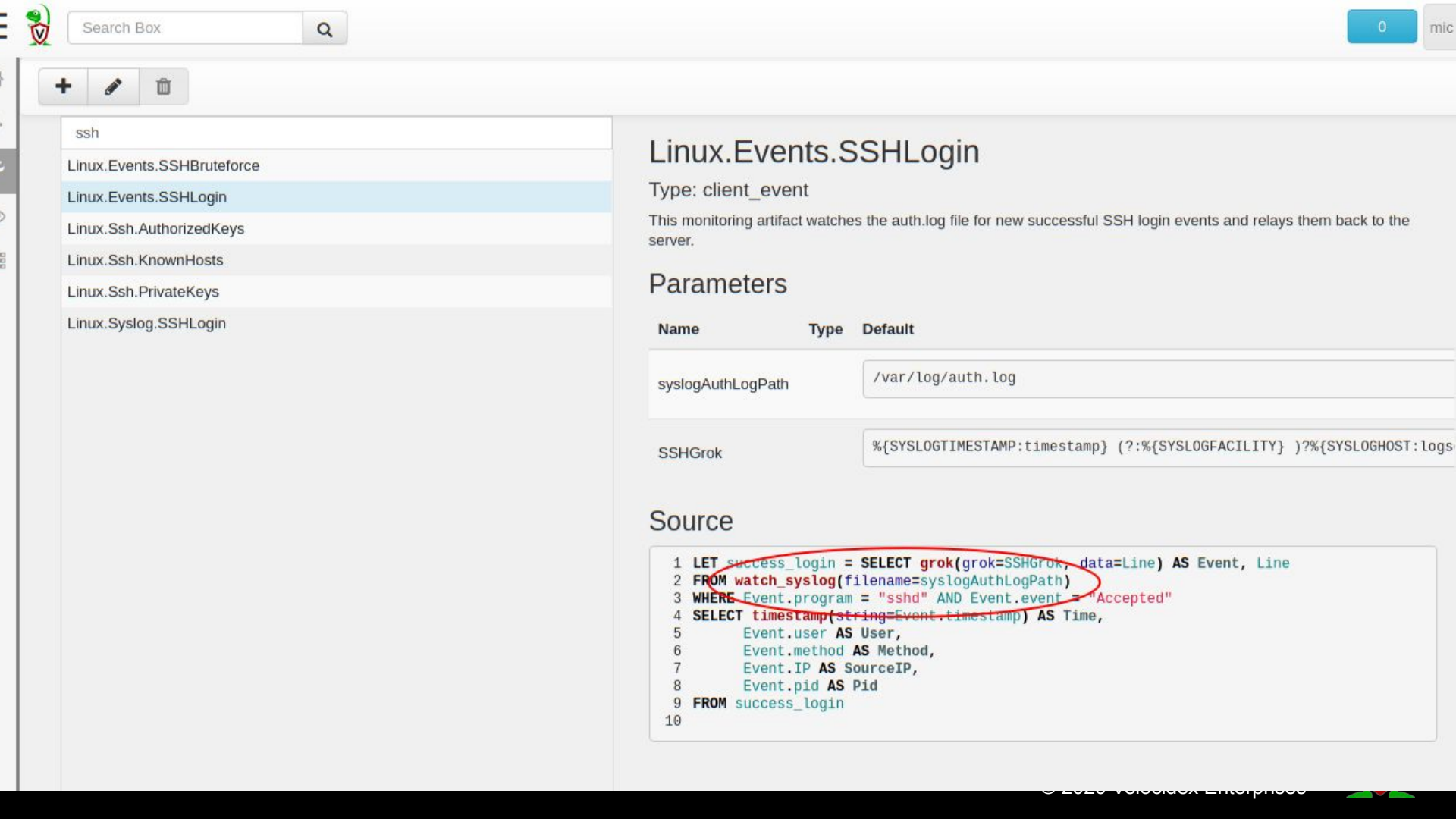
VQL Query

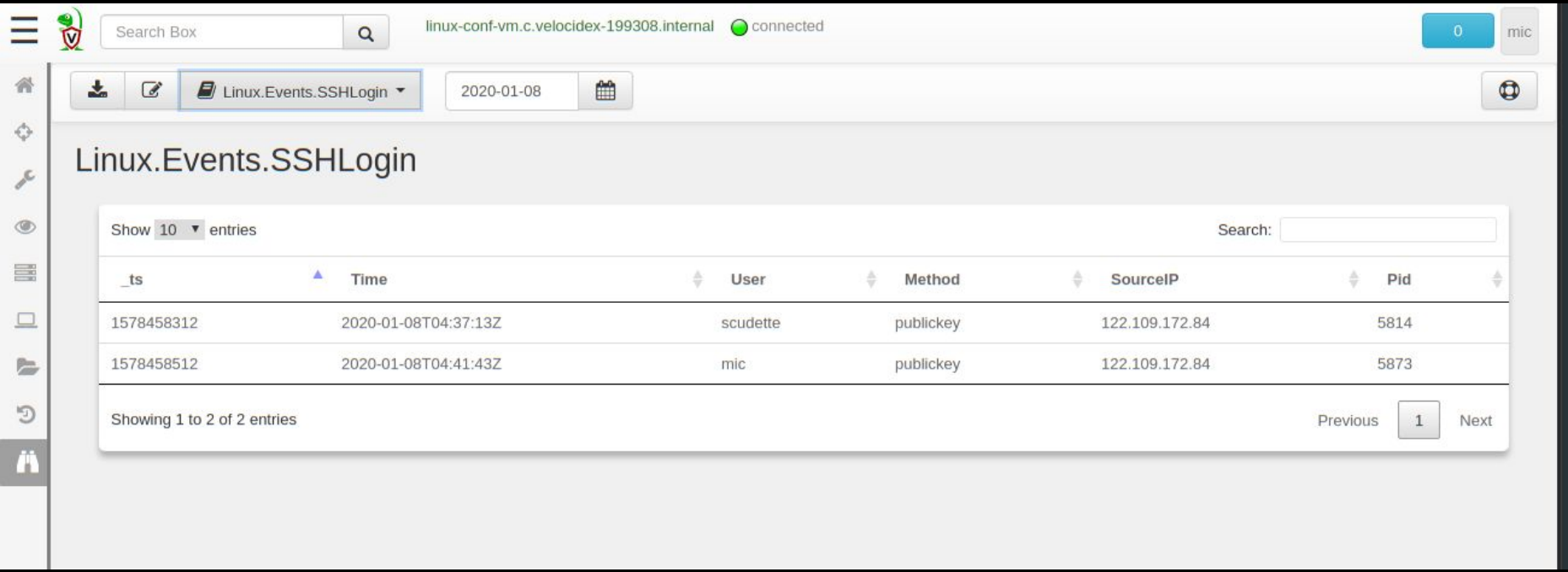
Scenario: SSH login

Monitor all endpoints for successful SSH connections.

As soon as a new SSH session starts, stream an event log to the server.

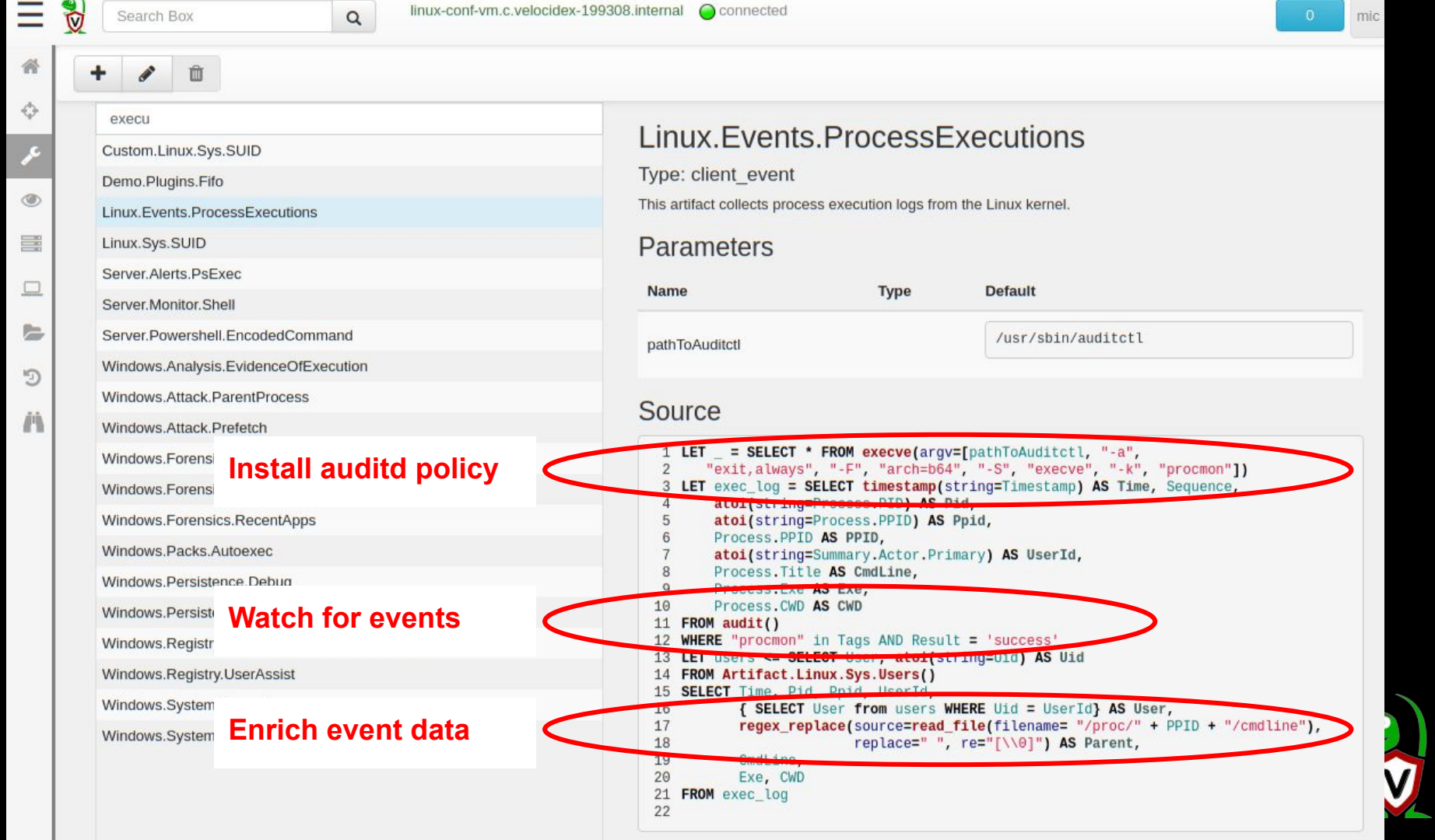
Now even if the attacker removes local log files we have this important information in a central location.

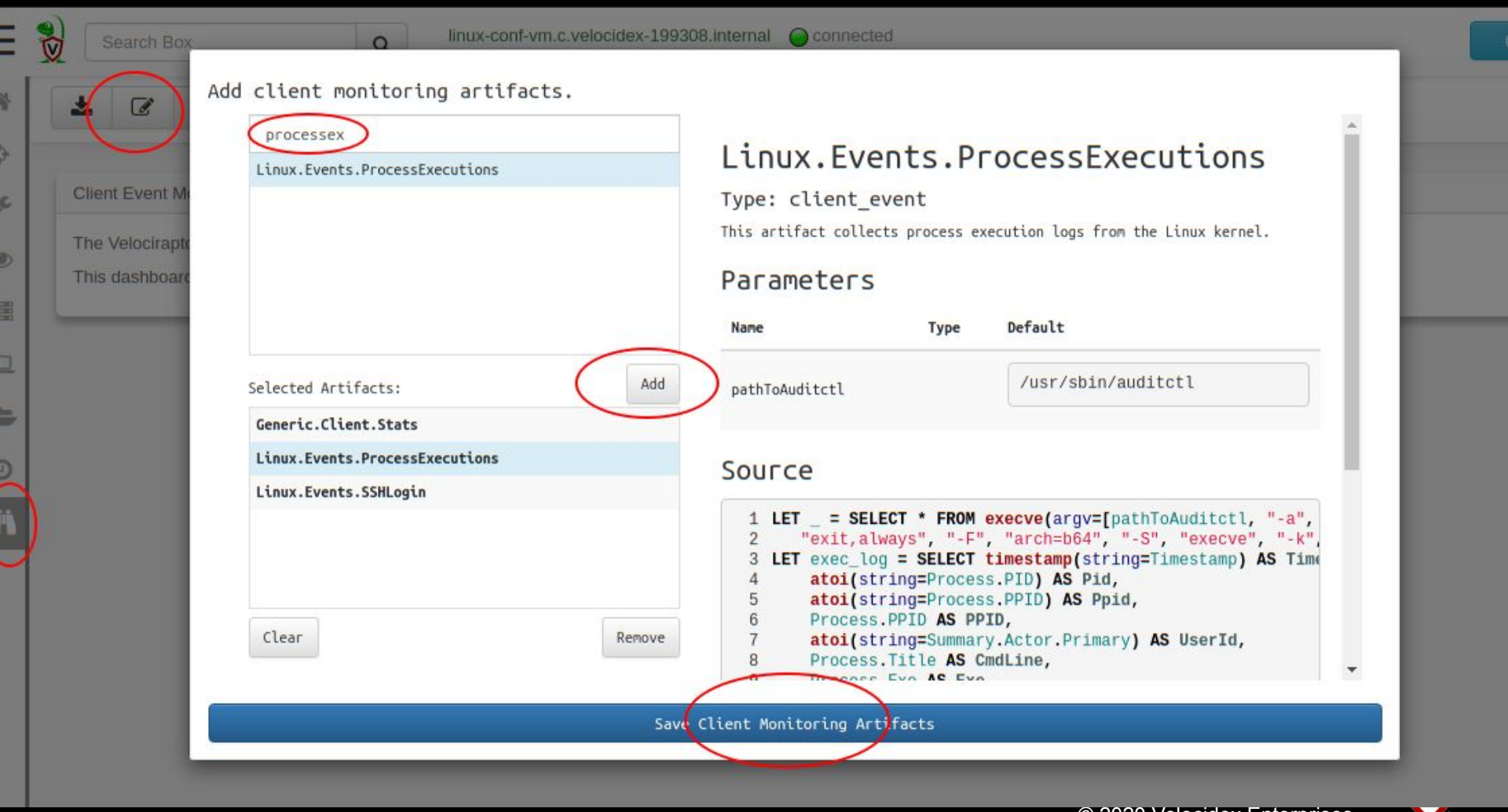

Search Box
0
mic
ssh
Linux.Events.SSHBruteforce
Linux.Events.SSHLogin
Linux.Ssh.AuthorizedKeys
Linux.Ssh.KnownHosts
Linux.Ssh.PrivateKeys
Linux.Syslog.SSHLogin
Linux.Events.SSHLogin
Type: client_event
This monitoring artifact watches the auth.log file for new successful SSH login events and relays them back to the server.
Parameters
Name
Type
Default
syslogAuthLogPath
/var/log/auth.log
SSHGrok
%{SYSLOGTIMESTAMP:timestamp} (?:%{SYSLOGFACILITY})?%{SYSLOGHOST:logs
Source
LET success_login = SELECT grok(grok=SSHGrok, data=Line) AS Event, Line
FROM watch_syslog(filename=syslogAuthLogPath)
WHERE Event.program = "sshd" AND Event.event = "Accepted"
SELECT timestamp(string=Event.timestamp) AS Time,
Event.user AS User,
Event.method AS Method,
Event.IP AS SourceIP,
Event.pid AS Pid
FROM success_login

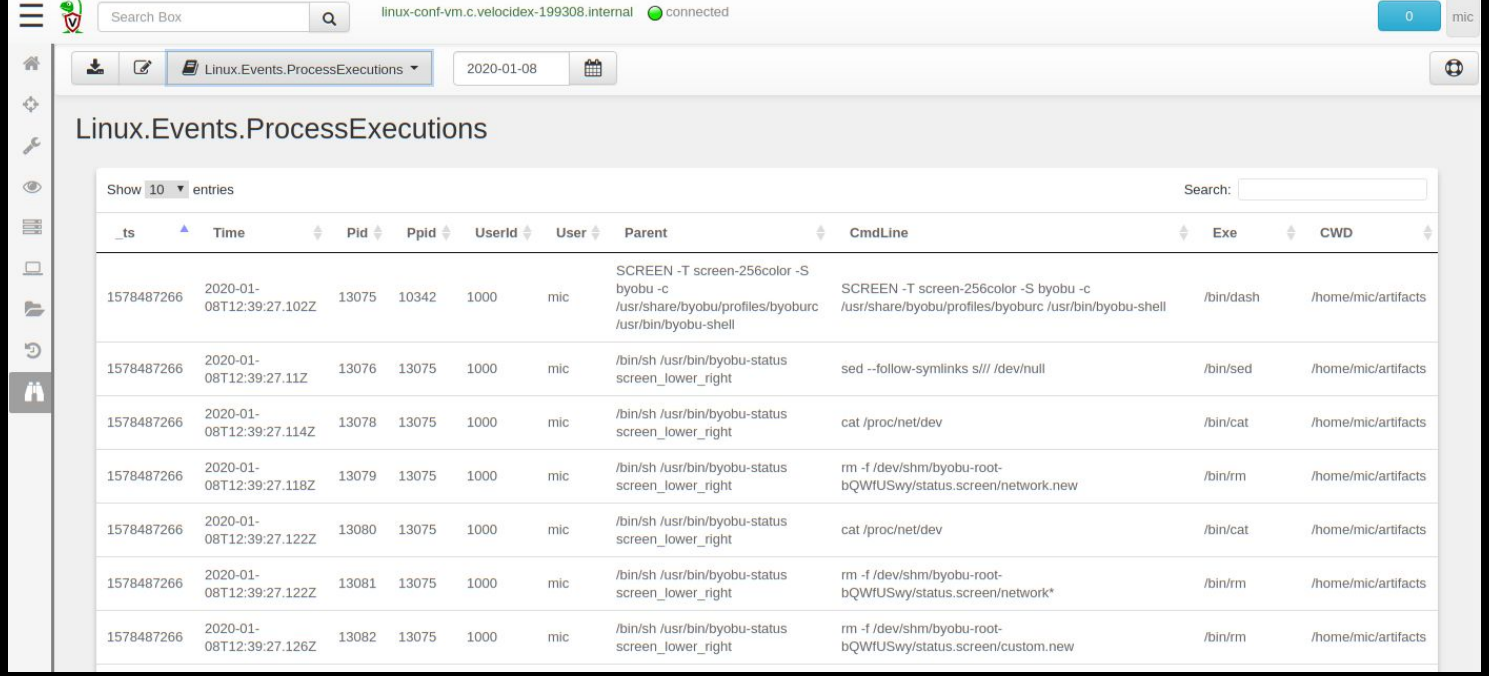


Process execution logs

- Collecting process execution is a very powerful technique
  - It is possible to isolate many anomalies, particularly for servers with predictable loads
- Examples:
  - Apache spawning bash -> execve vulnerability in PHP scripts or webshell
  - sshd spawning scp or sftp or rsync -> remote exfiltration of data
- On Linux process execution is logged by the kernel primarily via auditd.
- Velociraptor can parse auditd logs or act as an auditd broadcast receiver by itself.
  - Either way auditd must be installed in order to enable the audit rules


Search Box
linux-conf-vm.c.velocidex-199308.internal
connected
execu
Custom.Linux.Sys.SUID
Demo.Plugins.Fifo
Linux.Events.ProcessExecutions
Linux.Sys.SUID
Server.Alerts.PsExec
Server.Monitor.Shell
Server.Powershell.EncodedCommand
Windows.Analysis.EvidenceOfExecution
Windows.Attack.ParentProcess
Windows.Attack.Prefetch
Windows.Forensics.RecentApps
Windows.Packs.Autoexec
Windows.Registry.UserAssist
Linux.Events.ProcessExecutions
Type: client_event
This artifact collects process execution logs from the Linux kernel.
Parameters
Name
Type
Default
pathToAuditctl
/usr/sbin/auditctl
Source
LET _ = SELECT * FROM execve(argv=[pathToAuditctl, "-a",
"exit,always", "-F", "arch=b64", "-S", "execve", "-k", "procmon"])
LET exec_log = SELECT timestamp(string=Timestamp) AS Time, Sequence,
atoi(string=Process.PID) AS Pid,
atoi(string=Process.PPID) AS Ppid,
Process.PPID AS PPID,
atoi(string=Summary.Actor.Primary) AS UserId,
Process.Title AS CmdLine,
Process.Exe AS Exe,
Process.CWD AS CWD
FROM audit()
WHERE "procmon" in Tags AND Result = 'success'
LET users <= SELECT User, atoi(string=Uid) AS Uid
FROM Artifact.Linux.Sys.Users()
SELECT Time, Pid, Ppid, UserId,
{ SELECT User from users WHERE Uid = UserId} AS User,
regex_replace(source=read_file(filename= "/proc/" + PPID + "/cmdline"),
replace=" ", re="[\\0]") AS Parent,
CmdLine,
Exe, CWD
FROM exec_log
Install auditd policy
Watch for events
Enrich event data





_ts	Time	Pid	Ppid	UserId	User	Parent	CmdLine	Exe	CWD
1578487266	2020-01-08T12:39:27.102Z	13075	10342	1000	mic	SCREEN -T screen-256color -S byobu -c /usr/share/byobu/profiles/byoburc /usr/bin/byobu-shell	SCREEN -T screen-256color -S byobu -c /usr/share/byobu/profiles/byoburc /usr/bin/byobu-shell	/bin/dash	/home/mic/artifacts
1578487266	2020-01-08T12:39:27.11Z	13076	13075	1000	mic	/bin/sh /usr/bin/byobu-status screen_lower_right	sed --follow-symlinks s/// /dev/null	/bin/sed	/home/mic/artifacts
1578487266	2020-01-08T12:39:27.114Z	13078	13075	1000	mic	/bin/sh /usr/bin/byobu-status screen_lower_right	cat /proc/net/dev	/bin/cat	/home/mic/artifacts
1578487266	2020-01-08T12:39:27.118Z	13079	13075	1000	mic	/bin/sh /usr/bin/byobu-status screen_lower_right	rm -f /dev/shm/byobu-root-bQWfUSwy/status.screen/network.new	/bin/rm	/home/mic/artifacts
1578487266	2020-01-08T12:39:27.122Z	13080	13075	1000	mic	/bin/sh /usr/bin/byobu-status screen_lower_right	cat /proc/net/dev	/bin/cat	/home/mic/artifacts
1578487266	2020-01-08T12:39:27.122Z	13081	13075	1000	mic	/bin/sh /usr/bin/byobu-status screen_lower_right	rm -f /dev/shm/byobu-root-bQWfUSwy/status.screen/network*	/bin/rm	/home/mic/artifacts
1578487266	2020-01-08T12:39:27.126Z	13082	13075	1000	mic	/bin/sh /usr/bin/byobu-status screen_lower_right	rm -f /dev/shm/byobu-root-bQWfUSwy/status.screen/custom.new	/bin/rm	/home/mic/artifacts

Velociraptor simply reports process execution logs from the endpoint to the server

Advanced topics

- We can now forward these events to a SIEM like Elasticsearch
- We can write server monitoring artifacts to monitor the stream of events for bad patterns and alert/escalate further.

Apply what you learned

Investigations involve answering questions about our endpoints.

Velociraptor enhances your visibility into the state of your endpoints answering those questions effortlessly and quickly.

What questions do you want to answer about your network?

Dig Deeper!

- ❏ Today's workshop is an introduction level.
- ❏ Velociraptor's power comes from its open architecture and powerful query language.
- ❏ Learning how to write your own queries and integrate with Velociraptor's API opens huge possibilities for customization to your specific requirements!

See the documentation site Velociraptor Docs

Further training opportunities Training Schedule

Start hunting today

- ❏ Download Velociraptor from our GitHub repository.
- ❏ Join the mailing list at velociraptor-discuss@googlegroups.com
- ❏ Join our Discord Channel at https://www.velocidex.com/discord
- ❏ Leave feedback at http://feedback.velocidex.com/
- ❏ Contribute back with your feedback and ideas.