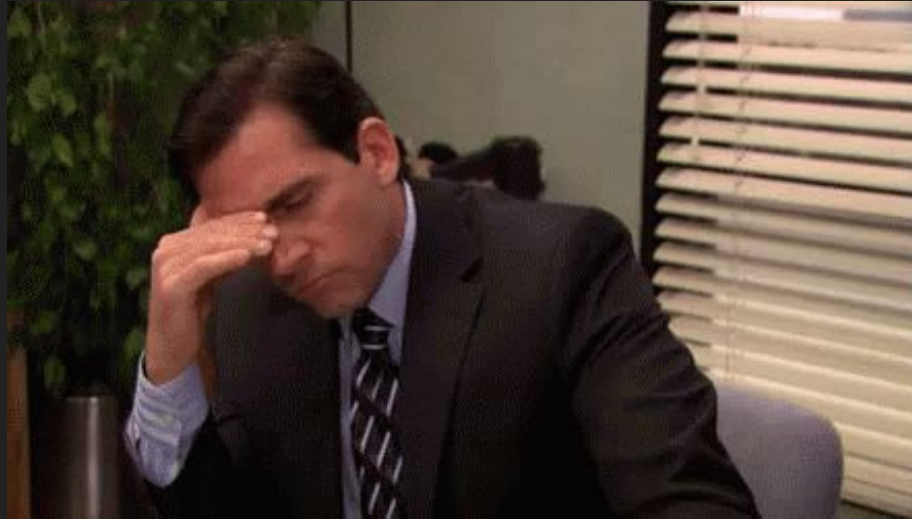


dfiq

DFIQ

Codifying Digital Forensics Intelligence

What even is “forensics”?



“Answering a question by uncovering digital traces and interpreting them in a reproducible way”

“Answering a question by uncovering digital traces and interpreting them in a reproducible way”

## That is the question

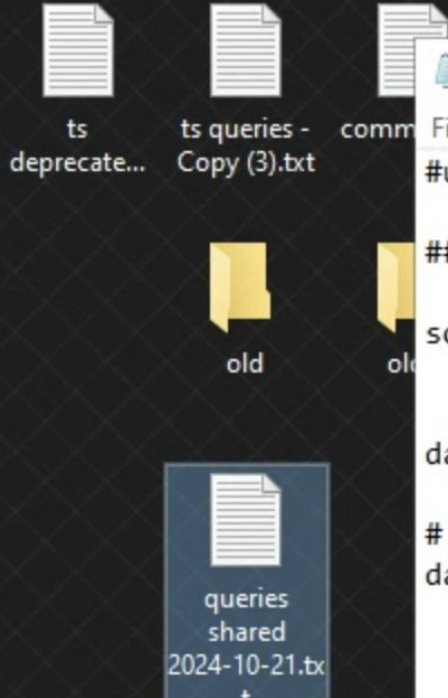
- High-level: “What happened on this computer?”
  - Low-level: “how did the malware persist?”
- Digital trace: “4624 from <IP> event found in Security.evtx”
- Interpretation: “Attacker logged onto the system”

“How did you find this?”

“oh, very simple... just ran the dftimewolf recipe manually specifying event logs in GRR and quick turbinia settings and then ran this cool query on the timesketch”

[🥲 understanding some of the words]  
“uuh... where’s the query?”





```
ts queries.txt - Notepad
File Edit Format View Help
#useful qureies

## login

source_name:Microsoft-Windows-Security-Auditing AND (event_identifier:4624)

data_type:"windows:evtx:record" AND event_identifier:4624

# old
data_type:"windows:evtx:record" AND event_identifier:"4624" AND username:*
```

We got a couple problems

- Knowing **what** questions to ask
- Knowing **where** to find “digital traces” (aka artifacts)
- Knowing how to **interpret** findings
- Answering questions **consistently**

new phone who dis?

Thomas Chopitea

[@tomchop](#)

DFIR @ Google

Yeti creator and core dev



new phone who dis?

Thomas Chopitea

[@tomchop](#)

DFIR @ Google

~~Yeti creator and core dev~~

✨ DFIQ evangelist ✨





## Get ready to be evangelized

- DFIQ in theory
- DFIQ in practice
- Open source & implementation challenges





dfiq

What DFIQ?

# Digital Forensics Investigative Questions



## Catalogue of **questions** and **how to answer them**

-  Make investigations consistent and explainable
-  Lower the barrier of entry to forensics
-  Distribute knowledge globally
-  Be system-agnostic (Yaml-based)

# Scenarios, Facets, Questions, Approaches...



- **Questions** have **Approaches** that describe how to answer them
- Questions are grouped into **Facets**
  - ~ question groups, e.g. “Persistence”
- Facets are grouped into **Scenarios**
  - ~ investigation types, e.g. “Host compromise”



## What's a good question?

- Generic enough to be reusable
  - ~~Was blah.exe downloaded by Chrome?~~
- Specific enough to be relevant
  - ~~What files were written to disk?~~
- Has a documented approach that can be used to answer the question

**QUESTION:**

***“What files were downloaded using a Web browser?”***

## What files were downloaded using a WB?

- Examine filesystem events in ~/Downloads
- Examine on-host browser history files
- Examine file writes originating from known browser processes

# Question

```
---
name: What files were downloaded using a web browser?
type: question
description:
uuid: 8620a183-d67f-481e-a63c-d8b8dfa5e968
id: Q1001
dfiq_version: 1.1.0
tags:
parent_ids:
- F1008
- F1002
approaches:
> - name: Detect browser downloads via change journal records ...
> - name: Detect browser downloads via file system event logs ...
> - name: Collect download records from local browser artifacts (Plaso) ...
> - name: Collect download records from local browser artifacts (Hindsight) ...
```

# Zoom-in: Approaches

```
- name: Collect download records from local browser artifacts (Plaso)
  description: Long description of what this approach looks like
  tags: [Web Browser, Chrome, etc.]
  references:
    - '[Chrome] (https://forensics.wiki/google\_chrome/#downloadsstart\_time)'
    - '[Web Browsers on ForensicArtifacts] (https://github.com/ForensicArtifacts)'
  notes:
    - [...]
  steps:
    - name: Collect ForensicArtifact data ...
    - name: Process data with Plaso ...
    - name: Filter the results to just file downloads ...
    - name: Filter the results to just file downloads ...
```

# Approaches: references & notes

```
· · references:
· · - · '[Chrome] (https://forensics.wiki/google\_chrome/#downloading)'
· · - · '[Web Browsers on ForensicArtifacts] (https://github.com/ForensicArtifacts/artifacts/blob/master/Windows/Browsers.md)'
· · notes:
· · · covered:
· · · - Chrome downloads. Beyond "stable" Chrome, this also includes
· · · - Safari downloads
· · · - Includes downloads from all Chrome profiles
· · · not_covered:
· · · - Firefox downloads
· · · - Downloads on any other browsers
· · · - Browsers installed in non-standard paths
· · · - Downloads made during Incognito sessions
```

## Approaches: steps

```
· steps:  
> · - name: Collect ForensicArtifact data ...  
> · - name: Process data with Plaso ...  
> · - name: Filter the results to just file downloads ...  
> · - name: Filter the results to just file downloads ...
```



## Zoom-in: steps - collection

dfiq

```
steps:  
- name: Collect ForensicArtifact data  
  description: Collect local browser hi  
  stage: collection  
  type: ForensicArtifact  
  value: BrowserHistory
```



## Zoom-in: steps - processing

```
-----  
- name: Process data with Plaso  
  description:  
  stage: processing  
  type: command  
  value: log2timeline.py --parsers chrome_history {path_to_evidence}
```

## Zoom-in: steps - analysis

```
- name: Filter the results to just file downloads
  description:
  stage: analysis
  type: opensearch-query
  value: data_type: ("chrome:history:file_downloaded"
    OR "safari:downloads:entry")
```

# Scenario

```
---
name: Suspicious DNS Query
type: scenario
description: >
  · A DNS query to an unexpected domain can be an indicator of abnormal activity on
  · a host. If a domain has been marked as malicious, an investigator may be tasked
  · with determining what caused the DNS query (or response) and if it indicates the
  · host has been compromised.
uuid: 5305f225-b274-4fc4-b60f-a87d9e2a7c11
id: S1003
dfiq_version: 1.1.0
tags:
- Network
- Malware
- Triage
```



How DFIQ do you use this?

## How to use?

- Not all approaches are relevant to all environments
  - e.g. custom workflows, proprietary systems
- Internal vs. public approaches
- System agnostic Yaml schema
  - Wiki generation (e.g. <http://dfiq.org>)

# “Are preload mechanisms being abused?”

dfir



## 🔍 Audit

- /etc/ld.so.preload:
- /etc/ld.so.conf
- /etc/ld.so.conf.d/\*.conf

## 🔍 Periodically check

/proc/{pid}/environ

## 🔍 Has anyone tampered with my loader (recent blog post on [dfir.ch](https://dfir.ch))?

```
# rpm -V glibc
```

[“In-Depth Study Of Linux Rootkits” on YouTube](#)

Stephan Berger

“Are there executables in odd locations?”

## SUSPICIOUS LOCATIONS

```
Windows.Search.FileFinder
SearchFilesGlobTable      Glob C:\Windows\System32\config\systemprofile\**\*.dll
                           C:\Windows\System32\config\systemprofile\**\*.exe
                           C:\Windows\System32\config\systemprofile\**\*.bat
                           C:\Windows\System32\config\systemprofile\**\*.ps1
                           C:\Windows\System32\config\systemprofile\**\*.cmd
                           C:\Windows\Tasks\*.dll C:\Windows\Tasks\*.exe
                           C:\Windows\Tasks\*.bat C:\Windows\Tasks\*.ps1
                           C:\Windows\Tasks\*.cmd C:\Users\**\*.dll
                           C:\Users\**\*.exe C:\Users\**\*.bat C:\Users\**\*.ps1
                           C:\Users\**\*.cmd C:\Windows\Temp\**\*.dll
                           C:\Windows\Temp\**\*.exe C:\Windows\Temp\**\*.bat
                           C:\Windows\Temp\**\*.ps1 C:\Windows\Temp\**\*.cmd
                           C:\Windows\*.cmd C:\Windows\*.exe C:\Windows\*.bat
                           C:\Windows\*.ps1 C:\Temp\*.cmd C:\Temp\*.exe
                           C:\Temp\*.bat C:\Temp\*.ps1 C:\*\*.dll C:\*\*.exe
                           C:\*\*.bat C:\*\*.ps1 C:\*\*.cmd C:\*.exe C:\*.dll
                           C:\*.bat C:\*.ps1 C:\*.cmd

Calculate_Hash             Y
```

[“The Gist of Hundreds of Incident Response Cases” on YouTube](#)

Stephan Berger  
(again lol)

dfiq

How DFIQ do we use it?





# It's been a whole year already??

dfiq

takeaways



- Yeti moves from a CTI platform to a **DFI platform**
- Acts as a automated, reusable **forensics KB**, leveraging **DFIQ**
- Helps forensic analysts automatically weed out the bad by providing ways to **slice and dice data**

*(Tom can't help it and keeps doing full software rewrites)*

# Yeti - Question & approaches



1. Collect ForensicArtifact data

ForensicArtifact

collection

NTFSUSNJournal

2. Process data with Plaso

command

processing

Plaso

3. Filter the results to file system rename events where the original file name ended with `.crdownload``.

opensearch-query

analysis

```
data_type:"fs:ntfs:usn_change" filename:crdownload "USN_REASON_RENAME_OLD_NAME"
```

4. Select and search for the `file_reference`` value for an event of interest from the previous query. There should be one with the same timestamp as your previous event and its `filename`` value is the download's final name.

opensearch-query

analysis

```
data_type:"fs:ntfs:usn_change" {file_reference value} "USN_REASON_RENAME_NEW_NAME"
```














## Covered

- Chrome downloads to an NTFS volume

## Not covered

- All other browsers.
- Downloads from other Chromium/Chrome-based browsers *might* be collected via this method, but testing is necessary to confirm.
- Downloads on file systems other than NTFS.
- Downloads that would be covered, but happened long enough ago that the USN Journal records that would show it have been deleted.

# Yeti - DFIQ tree

- ✓ ? What files were downloaded using a web browser? 
- >  Detect browser downloads via change journal records
- >  Detect browser downloads via file system event logs
- ✓  Collect download records from local browser artifacts (Plaso) 
  -  BrowserHistory ForensicArtifact
  -  Process data with Plaso command
    -  Plaso
  -  Filter the results to just file downloads opensearch-query
    -  `data_type:( "chrome:history:file_downloaded" OR "safari:downloads:entry" )`
  -  Filter the results to just file downloads pandas
    -  `query('data_type in ("chrome:history:file_downloaded", "safari:downloads:entry")')`
- >  Collect download records from local browser artifacts (Hindsight)

# Yeti - Approach coverage




## Coverage notes (3 / 4)

 Covered [+ ADD](#)

Chrome downloads. Beyond "stable" Chrome, this also includes Chromium, Chrome (Microsoft Edge, Brave, and Opera) on Windows, macOS, and Linux

Safari downloads

Includes downloads from all Chrome profiles

 Not covered [+ ADD](#)

Firefox downloads

Downloads on any other browsers

# Yeti - Approach steps



## Step 3

Step name

Filter the results to just file downloads

Type

opensearch-query



Type of step, e.g. command. Auto-populated with all already existing types.

Stage

analysis

Stage of the step, e.g. collection. Auto

Value

```
data_type:( "chrome:history:file_downloaded" OR "safari:downloads:entry" )
```

Description (optional)

Describe the steps in more detail if needed.



## Automation

- Reads scenario, unrolls DFIQ graph
- Reads **collection, analysis**-type steps
- e.g. launches GRR flows, queries Timesketch
- Displays results to analyst

✨ LLMs???



# Automation



The screenshot shows a Google Colab notebook interface. At the top, the title bar reads "[NEW] DFIQ investigative questions with LLM" with a star icon. Below the title bar is a menu with options: File, Edit, View, Insert, Runtime, Tools, Help, and a link for "All changes saved". On the right side of the title bar, there are icons for "Comment", "Share", a settings gear, and a user profile picture.

The main workspace contains a vertical list of code cells. Each cell is marked with a green checkmark and a "0s" execution time. The cells are numbered [20] through [25].

- Cell [20]: "Show code"
- Cell [21]: "Show code"
- Cell [22]: "Show code"
- Cell [23]: "Show code"
- Cell [24]: "Show code"
- Cell [25]: Contains Python code:

```
sig_platform.auth_oidc()  
dfiq_scenario = sig_platform.search_dfiq("GCE Compromise Assessment")
```

Below the code cells, there are two collapsed sections indicated by downward-pointing chevrons:

- "Obtaining DFIQ data from Yeti"
- "Query Yeti for DFIQ data"

At the bottom of the notebook, a status bar shows a green checkmark, "0s", and "completed at 1:03 PM".





## Challenges

pov: you finally try to use the  
standard you designed 2 years  
ago

## Challenges - tackled

- Understanding the schema was hard
- No one wanted to write Yaml, lol
- Contribution process unclear. Public? Private?

Switching IDs?

```
id: S1003
```

```
id: Q1001.10
```

- First shot at implementation was tricky (DFIQ 1.1)

S1003... what's that about

dfiq

“We want to federate the creation of DFIQ objects”



## Challenges - TODO

- Data **curation** - avoiding duplicates
- Hand holding *vs.* **investigative creativity**
- How do we convince the team to contribute?
- How do we convince **the public** to chime in?

## Takeaways

- DFIQ is a way to **catalogue** forensic **questions** and **approaches** to answering them
- Increase investigation **consistency**, onboarding **speed**, knowledge **sharing**
- Simple schema → powerful **automation avenues**
- Open source! Anyone can **contribute**
- <https://dfiq.org>, <https://yeti-platform.io>