Research Recap: How To Automate Malware Campaign Detection With Telemetry Peak Analyzer

By Jason Zhang, Stefano Ortolani, Giovanni Vigna

Published: 2021-11-11 · Archived: 2026-04-05 22:37:16 UTC

Cyber security threats have been growing significantly in both volume and sophistication over the past decade with no sign of a slowdown. Naturally, this has also been accompanied by an increased collection of threat telemetry data, ranging from detonation timelines to IDS/IPS detections. Telemetry data, typically represented by enriched time series, often contains underlying peak signals which in turn correspond to a few informative events: occurrences of malware campaigns, heavily used malware delivery vectors, commonly affected verticals, and even anomalies possibly revealing the presence of false positives. While all this information clearly holds tremendous value, mining these data sets can be expensive and complex. As a result, organizations often find it challenging to gain further insights of the underlying threat landscape even though they have access to the data.

Recently at VirusBulletin Threat Intelligence Practitioners' Summit (TIPs) 2021, we presented our latest research aiming to tackle the challenges discussed above: Telemetry Peak Analyzer is a statistical approach to detect malware campaigns as they happen by relying on telemetry data in an efficient and scalable manner.

Read on to get the key insights of the presentation. We'll provide an overview of the characteristics of typical threat telemetry data collected by VMware Threat Analysis Unit (TAU) before discussing the proposed Telemetry Peak Analyzer. We'll then evaluate the effectiveness of our approach by showcasing some typical examples of detected malware campaigns using the threat telemetry data generated from production followed by a brief discussion on the usage for the open-source package.

Threat Telemetry Data

Threat telemetry data can be seen as a composite time series. For a given attribute or dimension such as file type, a composite time series can be decomposed into individual time series. For example, a file detection composite time series could contain two individual time series such as PDF and Excel time series, as illustrated in Figure 1. As seen from the figure, even if there are some peaks from an individual time series, the composite time series doesn't show clear peaks.

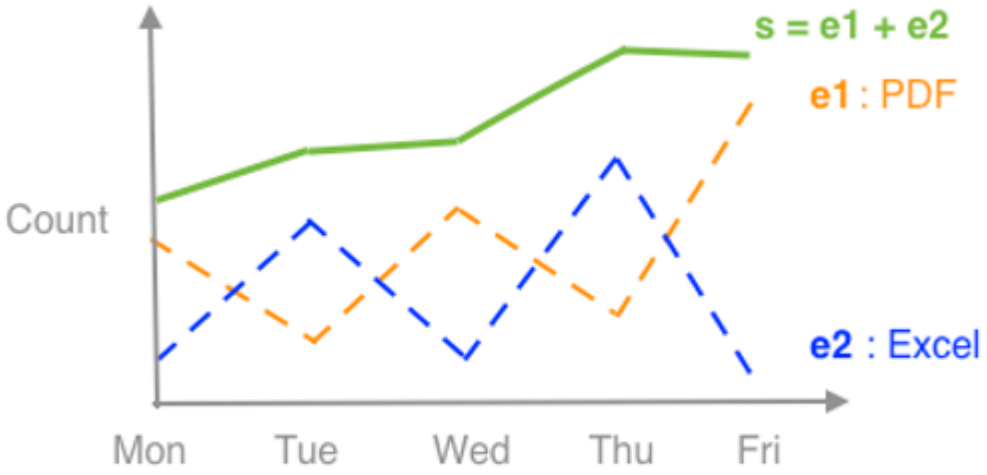

Figure 1: An illustration of a composite time series representing file detections.

Figure 2 shows [VMware NSX] threat telemetry data from production. The telemetry data is a composite time series that comprises multiple detection time series–one for each file type. While it is hard to tell if there are any malware campaigns (or underlying peaks) associated with certain file types by visually examining the telemetry data shown in Figure 2, you can identify an interesting peak. However, that peak is hidden.

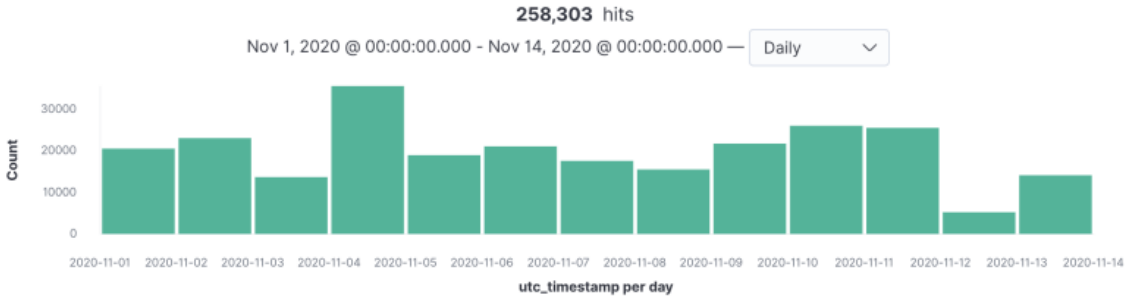

Figure 2: A snapshot of VMware NSX telemetry data comprising multiple detection timelines, one for each file type.

In addition, each individual time series has many detection events, and multiple detection events can refer to the same file. An event is further characterized by metadata (see Figure 3, some fields in the figure have been anonymized).

```
{
    "file.mime_type": "application/zip",
    "file.sha1": "fb9bd4c573cd3655250568f3bf969e6011bb1938",
    "submission_id": "1919216763",
    "analysis.label": "Trojan",
    "task.severity": "malicious",
    "file.llfile_type": "ZipArchiveFile",
    "source.data_center": "westus",
    "file.md5": "47d6d7eaf18f2a565f861e2212a79b73",
    "customer.sector": "Education",
    "source.access_key_id": 0,
    "source.geo.country_iso_code": "US",
    "task.portal_url": "https://user.lastline.com/portal#/analyst/task/4898d0ce90780010252557d802130ec5/overview",
    "file.magic": "Zip archive data",
    "customer.type": "commercial",
    "source.origin": "EMAIL",
    "file.sha256": "5a93c9a39d6739ece603a5b30189f9c176cf96065e4bef3b1870af6a435908ee",
    "file.name": "filename",
    "source.submitter_ip": "0.0.0.0",
    "task.score": "100",
    "source.geo.location": "0.00,0.00",
    "customer.region": "APAC",
    "customer.installation_type": "hosted",
    "customer.channel": "DIRECT",
    "utc_timestamp": 1602803064000,
    "source.user_id": 0,
    "file.size": "59696",
    "task.uuid": "4898d0ce90780010252557d802130ec5"
},
```


Figure 3: An example of file submission event from VMware NSX telemetry.

The telemetry data discussed above poses great challenges to malware campaign detection due to the complexity of the multi-dimensional time series.

Next, we introduce the proposed Telemetry Peak Analyzer.

Proposed Approach

The proposed Telemetry Peak Analyzer is a framework to analyze and detect peaks on telemetry data with composite time series. The analyzer detects meaningful peaks based on statistical measurements computed over a short local window and a longer global window of telemetry data:

- Local window: A short time data window in which we want to detect peaks of a given attribute or dimension, e.g., file type. During the detection process, the analyzer generates a local statistics table (LST) with all the necessary statistical measurements.
- Global window: A historical long time data window that serves as a global benchmark to determine if a detected peak within the local window is meaningful. During the detection process, it will generate (or update) a global statistics table (GST) with all the necessary statistical measurements.

Figure 4 illustrates the detection workflow overview of the analyzer. As pointed earlier, we define two data windows: a local window and a global window. The task is to identify legitimate peaks in the local window by leveraging the statistics in LST and GST tables through the peak detection algorithm.

After loading the statistics sets LST and GST, the peak detection logic proceeds by performing a set of statistical threshold-based comparisons. For instance, it compares the total event count in a local window of a given attribute (e.g., the file hashes of a given file type) to the weighted global average event count of the attribute in the

corresponding global window. A valid peak is detected after certain conditions are met. For the details of the detection algorithm, interested readers can refer to our VirusBulletin presentation or the open-source package.

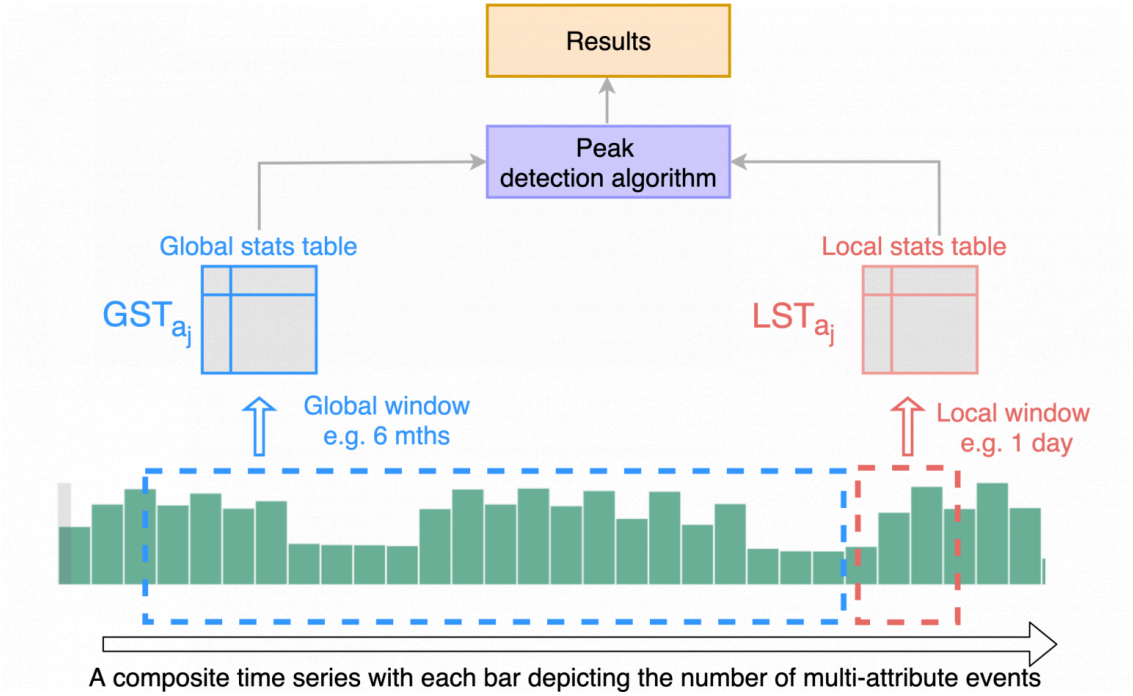

Figure 4: Peak detection workflow (first execution where tables are initialized and peaks analyzed).

It is worth noting that telemetry data is dynamic, therefore the global benchmark as reflected by GST needs to be updated over time. This is critical to maintain the accuracy of the peak analyzer over time. To make the global benchmark adaptive, the analyzer uses a sliding window mechanism to quickly update the new GST using previous GST and LST. This is to avoid re-calculating GST from scratch, thus greatly improving the performance of the algorithm. Figure 5 illustrates the process of updating the next global statistics table (denoted as GST^{i+1}) by using the data stored in GST and LST from the previous execution (denoted as GST^{i} and LST^{i} , respectively).

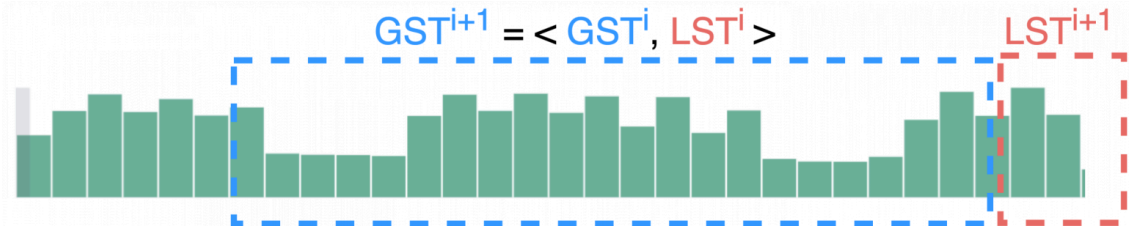

Figure 5: Subsequent executions updating LST and GST via a sliding window mechanism.

Evaluation

This section comprises two parts: Peak Detection and Performance.

In this part, we selected four different peaks as representative of the efficacy of the tool.

Phorpiex Peak

Phorpiex is a botnet which is known to distribute a range of malware such as BitRansomware. There was a Phorpiex campaign launched on November 4, 2020. The telemetry peak analyzer successfully detected the

malware campaign (see Figure 6) by detecting the surges from malicious Zip archive files from the telemetry data. The campaign mainly targeted universities in APAC region.

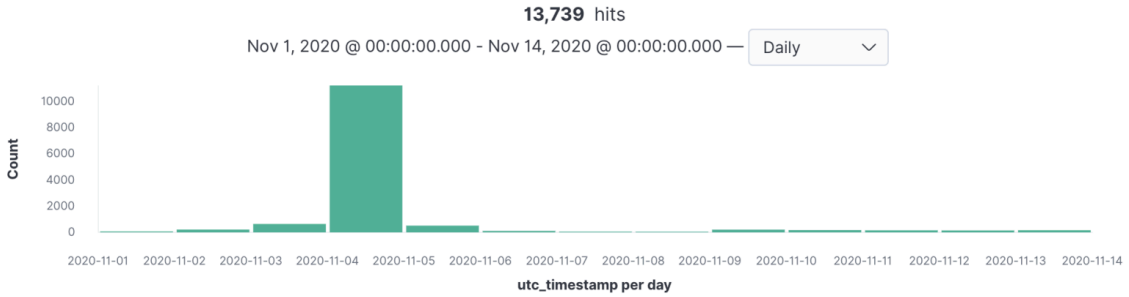

Figure 6: Phorpiex campaign detected by the surges from malicious Zip archive files.

More details on the Phorpiex campaign can be found in the report we published earlier: Phorpiex-Powered BitRansomware Targets APAC Universities.

Excel 4.0 Peak

Like VBA macros, Excel 4.0 (XLM) macros are a legitimate feature of Excel. These macros are increasingly abused by attackers to deliver a variety of malware families, such as Trickbot and Agent Tesla. The telemetry peak analyzer successfully detected the malware campaign (see Figure 7) by detecting the surges from malicious encrypted Excel files in the telemetry data.

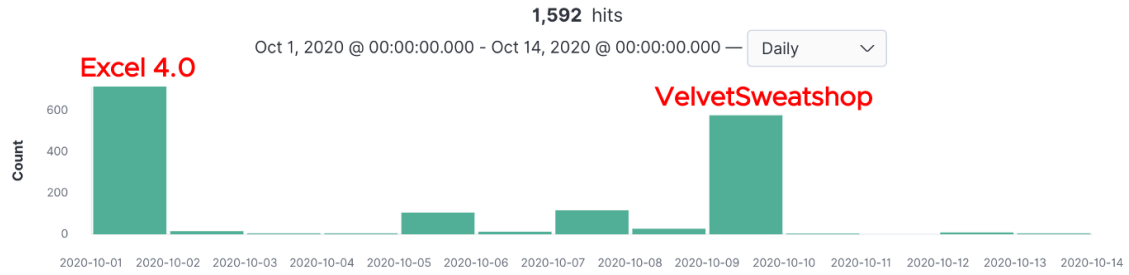

Figure 7: Excel 4.0 and VelvetSweatshop campaigns detected by the surges from malicious encrypted Excel files.

More details on weaponized Excel 4.0 macros can be found in the reports we published earlier: Evolution of Excel 4.0 Macro Weaponization and Evolution of Excel 4.0 Macro Weaponization – Part 2.

VelvetSweatshop Peak

VelvetSweatshop is a default encryption key stored in Microsoft Excel program code for decryption. Attackers leverage this trick to encrypt malicious Excel files in order to evade static-analysis-based detection systems, while eliminating the need for a potential victim to enter a password. The telemetry peak analyzer successfully detected the malware campaign (see Figure 7) by detecting the surges in encrypted Excel files from the telemetry data.

More details on the VelvetSweatshop campaign can be found in the report we published earlier: VelvetSweatshop: Default Passwords Can Still Make a Difference.

Another Excel 4.0 Peak

Recently VMware TAU detected another Excel 4.0 wave attacking our customers. As Figure 8 shows, the telemetry peak analyzer successfully detected surges from malicious Excel files in the telemetry data on October 20, 2021.

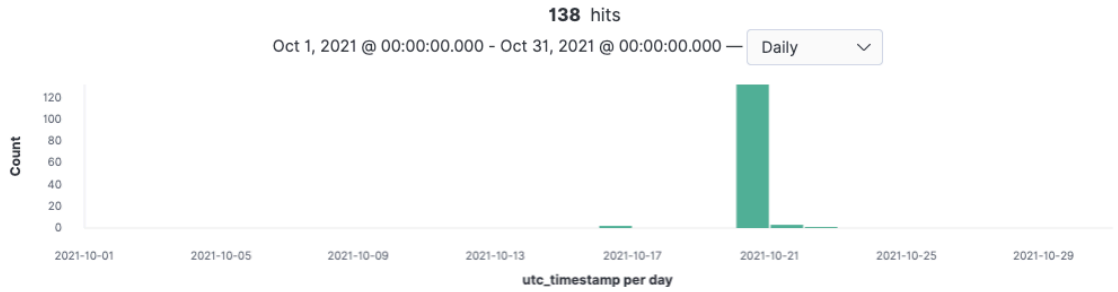

Figure 8: Excel 4.0 campaign detected by the surges from malicious Excel files.

The attacker leveraged highly obfuscated weaponized Excel 4.0 macros in the campaign. Figure 9 shows a code snippet embedded in one of the samples from the campaign (MD5: aa6fb794bbba6766a1cfb2474dfc3729).

```
Sub auto_open()
Excel4IntlMacroSheets.Add.Name = "Ciola"

Retio



End Sub

Function dfgdf()

Sheets("Ciola").Range("H24") = "h" & "tt" & "p" & ":/" & "/194.36.191.30/"
Sheets("Ciola").Range("H25") = "h" & "tt" & "p" & ":/" & "/23.106.122.40/"
dfgdf1
End Function
Function dfgdf1()
ddddddddd = "h" & "tt" & "p" & ":/" & "/94.140.112.52/"
Sheets("Ciola").Range("H26") = ddddddddd
Sheets("Ciola").Range("A1:M100").Interior.Color = vbBlack

Sheets("Ciola").Range("A1:M100").Interior.Color = vbBlack
End Function
```

Figure 9: Obfuscated malicious Excel 4.0 macros.

As highlighted in the figure, the code contains a few obfuscated URLs. Further investigation reveals that the macros attempted to download the very same payload from multiple URLs:

hxxp://194.36.191[.]30/44489.4081978009.dat

hxxp://23.106.122[.]40/44489.4081978009.dat

hxxp://94.140.112[.]52/44489.4081978009.dat

It seems that the attacker chose to use multiple C2 hosts to increase their chances to download the payload in case one or more hosts were taken down.

Exploring the sample and the URLs listed above on VirusTotal easily reveals similar samples and URLs from this campaign, as shown in Figure 10.

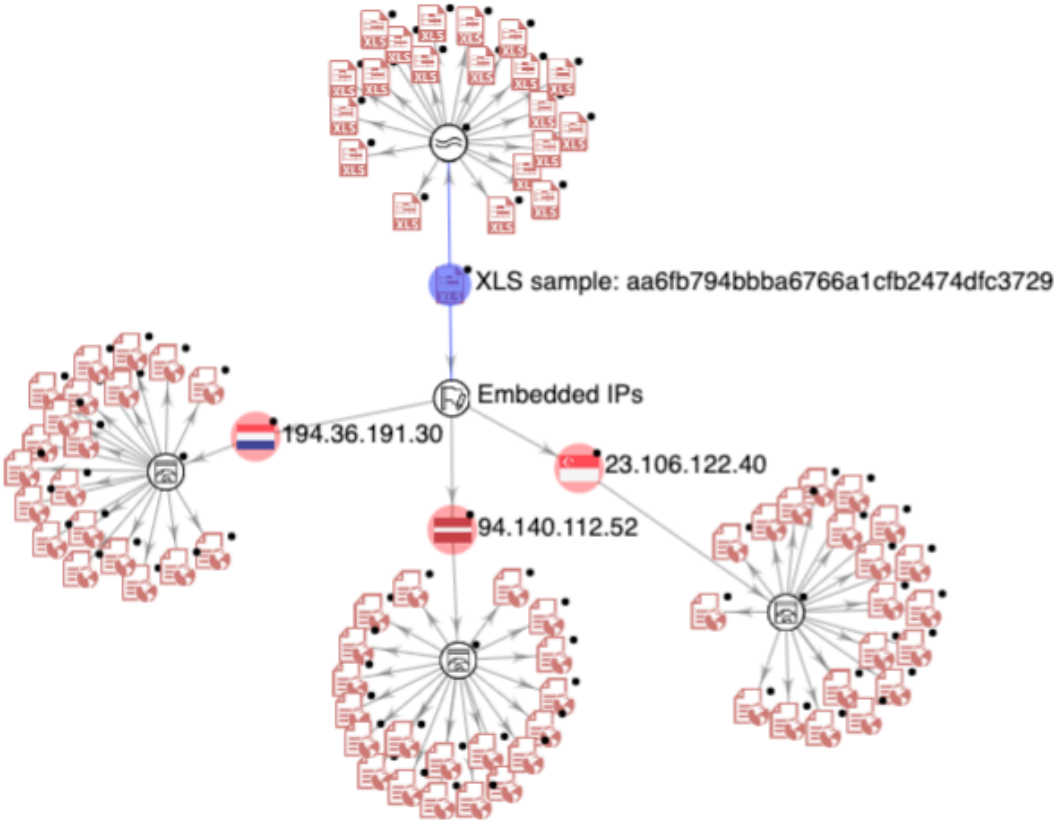

Figure 10: The correlation of indicators of compromise (IoCs) from this attack, created with VirusTotal Graph, visualizes the relationship between similar samples and the contacted hosts. The meaning of each node on the graph can be found here.

VMware NSX customers are well-protected against this kind of Excel 4.0-powered attacks. Figure 11 shows the analysis overview from a controlled environment when executing the initial malware. As shown, VMware's AI-driven NSX Advanced Threat Analyzer successfully detected a few high-risk artifacts, such as document executing regsvr32.exe, suspicious traffic, and potential social engineering attacks.

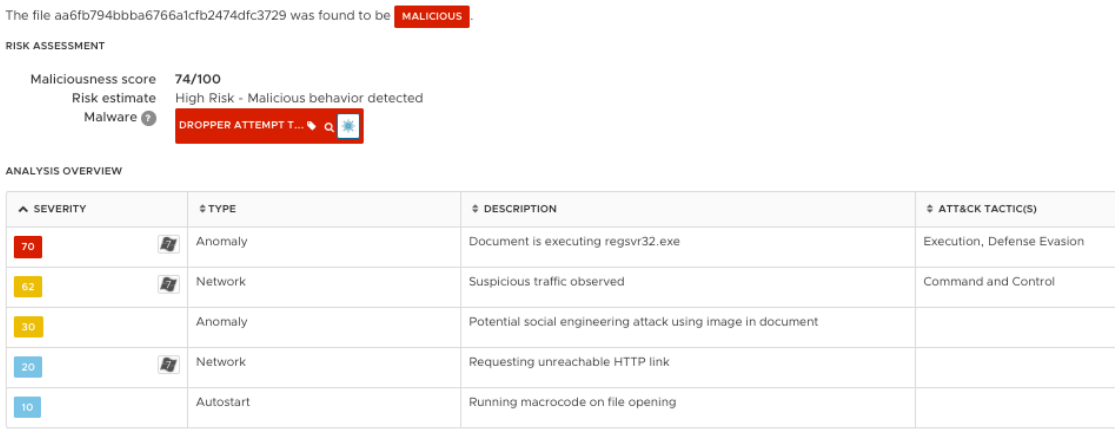

Figure 11: VMware NSX advanced threat analysis overview.

Currently, all the C2 URLs are not accessible. According to information shared by other researchers, the payload (MD5: aaf67a80f1dbb8406f14ca3f3af15ef7) is related to Qakbot (or Qbot), an infostealer that first surfaced as a banking Trojan in 2007. It is not the first time that attackers leveraged malicious Excel 4.0 macros to deliver Qakbot. Such attacks were also reported in February and May of this year.

Performance

Figure 12 shows the execution time and the number of peaks found in one of our datacenters since the end of December 2020. The first interesting property is that the execution time grows linearly with the number of peaks found, which is, in turn, proportional to the number of daily samples included in the time series (up to ≈ 4 million events per day in our experiments).

As shown in the figure, the Telemetry Peak Analyzer never took more than 90 seconds to complete, even when the number of legitimate peaks (including those arising from benign files) approached 15. This demonstrates the proposed Peak Analyzer can handle data at scale.

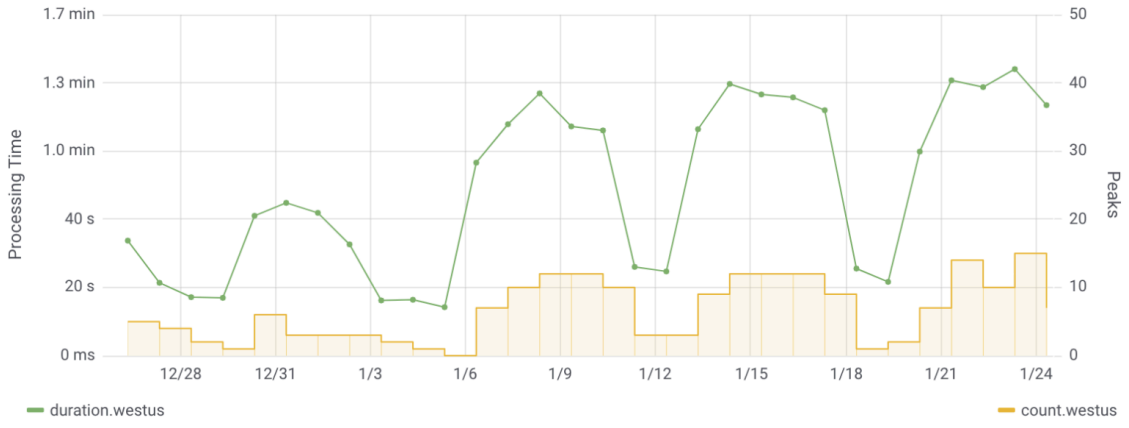

Figure 12: Processing times and identified peaks since end of December 2020.

The Open-Source Package

The Telemetry Peak Analyzer has become an open-source project:

Source Code

https://github.com/vmware-labs/telemetry-peak-analyzer

Usage

python -m telemetry_peak_analyzer -n ${input_json} -s ${start_time} -e ${end_time}

Example (from the repository root)

1. Run to generate global statistics table (GST):

python -m telemetry_peak_analyzer -n ./data/telemetry_example_3.json -s 2020-11-01 –e 2020-11-04

1. Run to simultaneously detect peaks and update the GST (in production repeated daily):

python -m telemetry_peak_analyzer -n ./data/telemetry_example_3.json -s 2020-11-04 –e 2020-11-05

The outputs from 1st run and 2nd run are shown in Figure 13 and Figure 14, respectively.

1st run output

```
(peak-analyzer) test@localhost telemetry-peak-analyzer % python -m telemetry_peak_analyzer -n ./data/telemetry_example_3.json -s 2020-11-01 -e 2020-11-04
INFO -> [2021-09-13 12:42:50] [0010mb] Loading Peak Analyzer from 2020-11-01 00:00:00 to 2020-11-04 00:00:00 with t=None
INFO -> [2021-09-13 12:42:50] [0010mb] Loading backend 'JsonBackend'
INFO -> [2021-09-13 12:42:50] [0010mb] Loaded files:
INFO -> [2021-09-13 12:42:50] [0010mb]  /Users/test/telemetry-peak-analyzer/data/telemetry_example_3.json
INFO -> [2021-09-13 12:42:50] [0010mb] Loading analyzer 'FileTypePeakAnalyzer' with backend 'JsonBackend'
INFO -> [2021-09-13 12:42:50] [0010mb] Loading global tables from file '/Users/test/telemetry-peak-analyzer/global_table.json'
INFO -> [2021-09-13 12:42:50] [0010mb]  Failed: [Errno 2] No such file or directory: '/Users/test/telemetry-peak-analyzer/global_table.json'
INFO -> [2021-09-13 12:42:50] [0010mb] Loading global tables from the backend
INFO -> [2021-09-13 12:42:50] [0114mb] Loading local tables
INFO -> [2021-09-13 12:42:51] [0131mb] Getting peaks
INFO -> [2021-09-13 12:42:51] [0131mb] Refreshing global tables
INFO -> [2021-09-13 12:42:51] [0131mb] Saving global tables to '/Users/test/telemetry-peak-analyzer/global_table.json'
```

Figure 13: Example output from 1st.

2nd run output

```
(peak-analyzer) test@localhost telemetry-peak-analyzer % python -m telemetry_peak_analyzer -n ./data/telemetry_example_3.json -s 2020-11-04 -e 2020-11-05
INFO -> [2021-09-13 12:42:59] [0011mb] Loading Peak Analyzer from 2020-11-04 00:00:00 to 2020-11-05 00:00:00 with t=None
INFO -> [2021-09-13 12:42:59] [0011mb] Loading backend 'JsonBackend'
INFO -> [2021-09-13 12:42:59] [0011mb] Loaded files:
INFO -> [2021-09-13 12:42:59] [0011mb]  /Users/test/telemetry-peak-analyzer/data/telemetry_example_3.json
INFO -> [2021-09-13 12:42:59] [0011mb] Loading analyzer 'FileTypePeakAnalyzer' with backend 'JsonBackend'
INFO -> [2021-09-13 12:42:59] [0011mb] Loading global tables from file '/Users/test/telemetry-peak-analyzer/global_table.json'
INFO -> [2021-09-13 12:42:59] [0011mb] Loading local tables
INFO -> [2021-09-13 12:43:00] [0116mb] Getting peaks
INFO -> [2021-09-13 12:43:00] [0116mb] TelemetryPeak(malicious, ZipArchiveFile) ...
INFO -> [2021-09-13 12:43:00] [0116mb] Refreshing global tables
INFO -> [2021-09-13 12:43:00] [0116mb] Saving global tables to '/Users/test/telemetry-peak-analyzer/global_table.json'
```

Figure 14: Example output from 2nd run.

Conclusion

It is important to identify hidden yet valuable information in threat telemetry data, such as detected malware campaigns and the customers and sectors that were affected the most. The proposed telemetry peak analyzer is designed to handle such composite time series telemetry data to effectively detect malware campaigns at scale, as demonstrated in the Evaluation part.

As part of future improvements, we plan to extend our prototype to process threat labels as well as dynamic behaviors. Currently we have separate peak analyzer instances running across different regions and feeds. In the future, we also plan to merge the state of the instances. As an open-source project, we encourage the open-source community to share feedback and help improve the project over time.
