Monitoring mobile device payments

Payments made through a mobile device, usually using a vendor partner to execute the payment for the underwriting bank, require analytics to ensure they are functioning as expected. Useful analytics include aggregates, locations, response times, and outliers.

Data required

Mobile device logs

Procedures

- Splunk recommends that customers look into using data models, report acceleration, or summary indexing when searching across hundreds of GBs of events in a single search. The searches provided here are a good starting point, but depending on your data, search time range, and other factors, more can be done to ensure that they scale appropriately.
- Some commands, parameters, and field names in the searches below may need to be adjusted to match your environment. In addition, to optimize the searches shown below, you should specify an index and a time range when appropriate.

► Mobile payment analytics

Payments by vendor

Summarizing the payment amounts per vendor over time is one way to judge performance. Using 95th percentile over time gives the underlying bank visibility into which partner is used for the most amounts for authorized purchases. You can see which vendor is in the lead at certain times of day and which ones are being used for the most payments, allowing for further analytic decisions to be made. Collect the time series data that contains mobile payment authorizations and what white label vendor they used, then run the following search.

```
| sourcetype=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort - _time
| search action="authorized"
| timechart fixedrange=F span=5m perc95(amount) BY vendor
```

Authorizations versus denials over time

Mobile payments, just like in-store payments, need to be monitored over time slices for authorizations and denials. Knowing how many authorizations and denials happen over a time period provides insights for you in near real-time to help improve efficiency, help with troubleshooting if there are outliers for a particular time slice, and help make decisions on how these outcomes are computed. For instance, it may be that some denials are too conservative and should have been authorized. By seeing a timechart that shows both trends in the same chart, you can then make
decisions with time series mobile payments data. Collect the time series data that contains mobile payment authorizations and denials for all customers, then run the following search.

```bash
| sourcetype=<mobile application logs> 
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S") 
| sort - _time 
| search action="authorized" OR action="denial" 
| timechart fixedrange=F span=1h count BY action
```

**Payments by operating system**

It is always relevant to know which mobile operating systems are used to make payments. This helps in development and optimization of mobile apps and also provides insight into sales and marketing opportunities. Collect the time series data that contains mobile payments that also contain the mobile OS used, then run the following search.

```bash
| sourcetype=<mobile application logs> 
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S") 
| sort - _time 
| chart count BY os 
| sort - count
```

**Payments and associated actions**

By calculating the increase or decrease of the entropy of the action value (which may be payments), you can measure if other values for other fields affects it. The decrease in entropy may make it easier to predict the value of action based on the value of something else in further machine learning use cases. For mobile payments, you can find out if certain locations, app versions, or operations decrease the entropy of the payment action. Collect the time series data that contains payment information, which may include the application version, location of payment origination, and the operation action (such as payment), then run the following search.

```bash
| sourcetype=<mobile application logs> 
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S") 
| sort - _time|associate improv=0.05 
| table Reference_Key, Reference_Value, Target_Key, Top_Conditional_Value, Description
```

**Mobile payment application troubleshooting**

**Operations associated with mobile payment application crashes**

By counting the operations running when a mobile payments app crashed, you can find out what operations the user was performing right before the crash. This helps troubleshoot which operations are most associated with crashes. Collect the time series data that contains payment information operations that are present when the mobile app for payments crashes, then run the following search.

```bash
| sourcetype=<mobile application logs> 
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S") 
| sort - _time 
| search action="crash"
```
Mobile payments errors by version and mobile OS

By mapping out which mobile devices, operating systems, and applications versions have the most errors, you can triage resources to fix those combinations that have the most counts. You can also see which ones have the least errors. Collect the time series data that contains payment information, the mobile OS used, and the version of the mobile app for the transaction, then run the following search.

| sourcetype=<mobile application logs>  
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")  
| sort -_time  
| search action="error"  
| stats count BY os, appVersion

Mobile payment application customer data

Most active mobile payment customers

By knowing which customers are making the most payments in a time span, it is easy to gauge which customers are most active. This provides opportunities for marketing, for understanding location-based activity, and for analyzing the success of incentives to make mobile payments. The data can also be used by security departments if there is an obvious outlier with too many payments. Collect the time series data that contains mobile payments by customers, then run the following search.

| sourcetype=<mobile application logs>  
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")  
| sort - _time  
| top limit=10 customer

Customers making the largest payments

Finding the highest value customers based on payment amounts for a time period has uses in marketing and sales. It serves as an indicator for how much money is being used by the top customers via mobile payments, and this can be compared to regular credit card payments to see if there is an upward trend for the use of mobile payments. Collect the time series data that contains mobile payment amounts for all customers, then run the following search.

| sourcetype=<mobile application logs>  
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")  
| sort - _time  
| search action="authorized"  
| stats sum(amount) AS sumAmount BY customer  
| sort - sumAmount  
| head 10  
| eval sumAmount=tostring(round(sumAmount, 2),"commas")

Mobile payment application help section usage

Help section clicks

The information provided in Splunk Lantern is intended for informational and educational purposes only. All information is provided in good faith, however, Splunk disclaims any and all representations and warranties, express and implied, regarding the information provided, including without limitation any warranties and representations regarding the completeness, adequacy or accuracy of the information. You agree to take full responsibility for the results arising from the use of the information provided.
Knowing which help sections of the mobile app were clicked on most can help you make improvements. This information can tell the docs team and developers which area needs better documentation or UI updates. Collect the time series data that contains the help sections clicked on the mobile payments app, then run the following search.

```
| sourcetype=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort - _time
| top limit=10 helpSection
```

**Average response time per help section**

Having fine-grained views on the average response time for different parts of the mobile payments app gives a clearer understanding of where to focus efforts to fix issues. It is one thing to know that the customer has clicked on a help section of the mobile payments app, but it is another to know the average response time for this section. Slower average response times lead to less customer satisfaction. Collect the time series data that contains the average response time for each help section of the mobile payments app, then run the following search.

```
| sourcetype=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort - _time
| timechart span=1h fixedrange=F avg(responseTime) AS avgResponseTime BY helpSection
```

**Mobile payment application response times**

**Response time per location**

By knowing the average response time per location, you can learn how each location is performing with the payment app. Sometimes distance or a bad network plays into high response times. This is important to understand if a particular location is causing problems so that network providers can address it quickly before customers start complaining. Collect the time series data that contains mobile payment response times and customer location, then run the following search.

```
| sourcetype=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort - _time
| timechart span=5m fixedrange=F avg(responseTime) AS AverageResponseTime BY location
```

**Response times per mobile OS**

By knowing the average response time per mobile OS, you can learn how each mobile OS is performing with the payment app. This is important to understand if a particular OS is causing problems so that an app developer can address it quickly before customers start complaining. Knowing the average response time over time for each mobile OS gives the underlying bank visibility into how each mobile OS is performing with the payment app. Spikes in the data show that either there was a flurry of activity, or OS or network issues that caused the average response time to slow down at certain times. Collect the time series data that contains mobile payment response times and what mobile OS they used, then run the following search.

```
```
Response times per vendor

By knowing the average response time per the partners payment channel, you can decide who has met and not met SLAs. This is important for promoting what vendor to use and to understand which vendors are having trouble meeting the expected response times. Knowing the average response time over time for each vendor partner gives the underlying bank visibility into how each payment partner is performing. Spikes in the data show that either there was a flurry of activity, or network issues that caused the average response time to slow down at certain times for a vendor. Collect the time series data that contains mobile payment response times and what white label vendor they used, then run the following search.

```
| source=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort - _time
| timechart span=5m fixedrange=F avg(responseTime) AS AverageResponseTime BY os
```

Response times without outliers

Sometimes outliers, that do not repeat, get in the way of real analytics. They can skew the numbers and create confusion. By removing them from the average response time, you can get a picture of what the time should be. Knowing the average response time over time for each mobile payment helps in planning and preventing customer dissatisfaction. Collect the time series data that contains mobile payment response times, then run the following search.

```
| source=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort - _time
| timechart span=5m fixedrange=F avg(responseTime) AS AverageResponseTime BY vendor
```

Response time outliers

Finding response times that are the highest using customer and mobile operating system combinations may help narrow down a Service Level Agreement problem and assist with troubleshooting, as it may be a particular mobile OS that is having trouble with responses. Collect the time series data that contains response times and mobile OS used for all customers, then run the following search.

```
| source=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort - _time
| timechart span=5m fixedrange=F avg(responseTime) AS AverageResponseTime
| outlier action=remove
```

The information provided in Splunk Lantern is intended for informational and educational purposes only. All information is provided in good faith, however, Splunk disclaims any and all representations and warranties, express and implied, regarding the information provided, including without limitation any warranties and representations regarding the completeness, adequacy or accuracy of the information. You agree to take full responsibility for the results arising from the use of the information provided.
Another way to find outliers is with the `anomalousvalue` command, which computes an anomaly score for each response time of each event relative to the values of this field across other events. As this is done in one command, outliers can be found easily leading to quicker MTTR for issues. Collect the time series data that contains response times and original location for all customers, then run the following search.

```
| sourcetype=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort - _time
| anomalousvalue responseTime
| eval amount=tostring(round(amount, 2),"commas")
```

**Unexpected response times by location**

Finding events that are not expected based on response time and location groupings may assist in troubleshooting. The *unexpectedness score of an event* is calculated based on the similarity of that event’s response time/location grouping to a set of previous events. Unexpected response times may be caused by the application, location, network, or a host of other details. Collect the time series data that contains response times and original location for all customers, then run the following search.

```
| sourcetype=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort - _time
| anomalies field=responseTime BY location
| table _time action customer location os vendor amount responseTime unexpectedness
| eval amount=tostring(round(amount, 2),"commas")
```

**Average response time of all events**

Comparing the average response time for the mobile payments app in time slices against the overall average response time of all events within the same time chart provides a visual on the upward or downward ebbs and flows of the overall system. When the average response time for a time slice goes over the overall average by a few standard deviations, it may be time to investigate and also see if the change affects the business outcome for mobile payments. Collect the time series data that contains to the average response time for each request from the mobile payments app, then run the following search.

```
| sourcetype=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort - _time
| eventstats avg(responseTime) AS avgResponseTime
| timechart span=5m fixedrange=F avg(responseTime) AS avgResponseTime avg(avgResponseTime) AS AvgAllEvents
```

**Average response time with trendline**

An upward or downward trend line can indicate changes in system performance that may lead to faster reactions for improvement, if necessary. By adding a moving average line with the original data, you can discover upward or
downward trends. Collect the time series data that contains mobile payments along with response times, then run the following search.

```
| sourcetype=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort - _time
| timechart avg(responseTime) AS avgResponseTime span=1h fixedrange=F
| trendline sma5(avgResponseTime) AS MovingAverage5Count
```

---

**Mobile payment performance and usage correlations**

**Operating system and location**

Knowing which locations are using which mobile operating system is an important statistic for grouping users and troubleshooting. For instance, if you find that users in City A are using OS B and have the worse satisfaction, then it's time to investigate whether the OS is causing application issues. You can also use this information to prepare to meet demand for future app versions. Collect the time series data that contains payment information, the mobile OS used, and the location of the mobile app for the transaction, then run the following search.

```
| sourcetype=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort - _time
| search action="authorized"
| contingency os location
```

Another way to run this search is with the `geostats` command. Collect the time series data that contains authorized events, including the mobile OS used, the location, and the amount, from the mobile payments app submitted from the customer, then run the following search.

```
| sourcetype=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort - _time
| search action="authorized"
| lookup CityLocations location
| geostats avg(amount) BY os latfield=latitude longfield=longitude binspanlat=8 binspanlong=8
```

**Vendor and operating system**

By understanding which white label vendor and mobile system combination is involved in the most or least payment amounts, you can further delve into the reasons for the outliers and take any necessary corrective or reinforcement action. Further investigation can be made to understand why a particular vendor and OS is at the top or bottom. Collect the time series data that contains authorized mobile payments, amount submitted, white label vendor used, and the mobile OS used, then run the following search.

```
| sourcetype=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort - _time
| search action="authorized"
```
Location and vendor

A quick, tabular view of mobile payment amounts for location and vendor combination gives a summary view of the business to see which location or payment vendor is doing best. This information can be used to compare with historical trends and to see which vendor location combination is most popular. Collect the time series data that contains mobile payment locations and vendor used for all customers, then run the following search.

```
| sourceType=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort - _time
| search action="authorized"
| stats sum(amount) AS Total_Payments BY location, vendor
| xyseries location, vendor, Total_Payments
| eval t=1
| addcoltotals|sort - t
| eval location=if(t>1, "Totals", location)
| fillnull MoonPay ButterCupPay SplunkPay
| fields - t AppVersion customer epoch operation os responseTime
| table location *
```

► Mobile payment average amounts and outliers

Payment amount outliers

Anomalies in mobile payments can include outliers outside of degrees of standard deviation, which can help in use cases for marketing, security, fraud, and troubleshooting. For instance:

- Faulty software may be sending less than what was actually paid because the user has an old version of a buggy app.
- The same customer might always show up as an outlier in batches of 1,000 payments compared. That same customer may not only be an outlier, but also in lock step with the same high payments every few minutes, which could indicate a bot has taken over their payment app.
- It can find trends for high and low value customers.

Collect the time series data that contains payment information amounts for all customers, then run the following search.

```
| sourceType=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| search action="authorized"
| streamstats window=100 avg(amount) AS avg stdev(amount) AS stdev
| eval lowerBound=(avg-stdev*2)
| eval upperBound=(avg+stdev*2)
| eval isOutlier=if(amount < lowerBound OR amount > upperBound, 1, 0)
| sort - isOutlier
| where isOutlier>0
| fields - epoch authorized isOutlier avg stdev
```
Payments amount outliers using average plus standard deviation

Finding customers that paid the most in a time period on average is an easy way to find high value customers. You might want to give them status, providing further marketing opportunities for mobile payments. Collect the time series data that contains payment information amounts for all customers, then run the following search.

```bash
| sourcetype=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort - _time
| search action="authorized"
| stats avg(amount) AS averageAmount BY customer
| eventstats avg(averageAmount) AS avgAll stdev(averageAmount) AS stdevAll
| where averageAmount>(avgAll+1.85*stdevAll)
| eval averageAmount=tostring(round(averageAmount, 2),"commas")
```

Payments amount outliers using IQR

The interquartile range (IQR) is a stable measure of spread in that it is not influenced by unusually large or unusually small values for payment amounts. This keeps one or two extreme values from influencing the whole data set and makes it easier for you to target ranges for further work involving marketing, troubleshooting, or finding commonalities that would otherwise be missed. Collect the time series data that contains payment information amounts for all customers, then run the following search.

```bash
| sourcetype=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| search action="authorized"
| eventstats median(amount) AS median p25(amount) AS p25 p75(amount) AS p75
| eval IQR=(p75-p25)
| eval lowerBound=(median-IQR)
| eval upperBound=(median+IQR)
| eval isOutlier=if(amount < lowerBound OR amount > upperBound, 1, 0)
| where isOutlier>0
| fields - epoch authorized isOutlier IQR median p25 p75
| eval amount=tostring(round(amount, 2),"commas")
```

Average payment clusters

Performing an average of payments over time does not give subtle information on what numbers comprise the average because one number can skew the average. Breaking it into categories, such as demographics, locations, and mobile operating system devices, can provide more insight. You can then use this information for building machine learning models to make the system more efficient and profitable. Collect the time series data that contains payment information amounts for all customers, then run the following search.

```bash
| sourcetype=<mobile application logs>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort - _time
| kmeans k=4 amount
```
Mobile payment predictions

Future counts

Predicting the future count of mobile payments provides a forecast for marketing and sales, a capacity number for network engineers, and an idea of future scale for application developers. Using the current count of authorized payments, you can use an elementary machine learning technique to calculate the estimated counts for the immediate future. Collect the time series data that contains authorized mobile payments, then run the following search.

```plaintext
| sourcetype=<mobile application logs> | eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S") | sort - _time | search action="authorized" | timechart count span=1h fixedrange=F | predict count future_timespan=2
```

Response times

Predicting the future average response time for mobile payments provides a sense of the capacity of the mobile payments system for networking engineers and application developers. Using the current average response time, you can use elementary machine learning to calculate the average response time for the immediate future. This helps in capacity planning and also gives an indication if past behavior may cause poor average response times in the future. Collect the time series data that contains authorized mobile payments and response times, then run the following search.

```plaintext
| sourcetype=<mobile application logs> | eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S") | sort - _time | search action="authorized" | timechart avg(responseTime) AS avgResponseTime span=1h fixedrange=F | predict avgResponseTime future_timespan=2
```

Least probable events in mobile payments

Anomalies in mobile payments can either be ignored (as in a marketing use case) or may stand out (as in a fraud use case). Either way, knowing which events are outliers can help analysts in investigations for SLA infractions, fraud attempts, and future marketing. Collect the time series data that contains payment information amounts for all customers, then run the following search.

```plaintext
| sourcetype=<mobile application logs> | eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S") | sort - _time | anomalydetection | fields - epoch | eval amount=tostring(round(amount, 2),"commas")
```
Next steps

Add the results of these searches to dashboards so you can monitor activity. Investigate requests and response times that do not meet your SLAs. You can also use the results to report on key performance indicators, improve your payment response application, and make decisions on how to improve customer service.

The Splunk Essentials for the Financial Services Industry app helps you automate the searches provided in this article. The app also provides more insight on how they can be applied in your environment, how they work, the difficulty level, and what data can be valuable to run them successfully. In addition, the Splunk Essentials for the Financial Services Industry app provides a number of other fraud detection solutions for financial services.

Finally, these additional Splunk resources might help you understand and implement this use case:

- .Conf Talk: 40 ways to use Splunk in financial services
- Webinar: AIB: Monitoring payments with real time insights using Splunk and ITSI