Predicting Security Threats with

splunk>

1. Getting to know SPLUNK

Predicting Security Threats with Splunk

1. Getting to Know Splunk

Hacking Wisdom

Table of Contents

Foreword

As the complexity of organizations increases, new challenges arise when it comes to preventing security threats.

There is an undiscussed need for new proactive approaches in detecting potential security threats, to complement the traditional static analysis; in fact, threat signals might be intercepted by simply listening to network traffic, putting in place Introusion Detection Systems, stateful inspection, etc, by implementing appropriate hardware and software solutions.

However, in order to gain a complete and wider picture about what’s going on, we need a different “holistic” mindset, achieved by leveraging on heterogeneous sources of information: that’s what Big Data solutions, such as Splunk, are meant for.

Let’s delve deeper.

Chapter One

Big Data in a Nutshell

You may have already heard of *Big Data*, but probably not in relation with IT Security: although Big Data is often associated with social media, actually it is much more than that; so let's recall some concepts regarding Big Data Analytics, before looking at their implementation in *Splunk.*

Conventionally, Big Data are characterized by *the three Vs: Volumes,Velocity, and Variety.*

- *Volume of data* (in terms of data storaging magnitude): the order of magnitude of data is usually considered starting from Petabytes to Zettabytes (and over). As more and more data are generated on a daily basis, *Volume* seems not to be a complete and appropriate criterion to tell Big Data apart from other "normal" data: as a matter of fact, as storage capabilities increase, along with their costs falling down, the Volume magnitude factor shifts constantly upwards;
- *Velocity of data:* is the speed at which data are generated or, on the other hand, the frequency at which data are delivered to organizations. As new hardware devices are put in place (sensors, cameras, rfid, even click stream of web sites, etc), the stream of data at our disposal flows at higher rate then ever before, demanding for a real time approach in getting rid of informations hidden in data streams.
- *Variety of data* , has not only to do with different data sources, but also with their inherently *unstructured* format: it is controversial if *data* could be at all considered as unstructured as *per their nature,* or if it is instead the lacking of information about metadata that leads to unpredictable models.

Nevertheless, even when we try to analyze well known textual objects, such as emails (that are structured in compliance with RFC-2822, for example), we cannot be sure if they come in a predefined model or schema (for example, we cannot predict how long the email *body field* would be, or if a *Subject field* would be present or not, and so on).

The aforementioned characteristics may be summarized into a single concept: the need for *scalability*.

In fact, what really distinguish Big Data from Relational databases, for example, is that the latter don’t scale very well when the magnitude of data increases exponentially, or when data streams produced by hardware appliances (over)flow in real time fashion, or data themself don’t seem to comply with predefined and *rigid* schemas.

To work correctly, Relational databases need predefined data models with clearly defined fields in associated tables; in a very sense, Relational database conform to *User driven* data analytics, implying the so-called *Early Structure Binding* of data; in other words, the analyst needs to know in advance what kind of questions are to be answered by analyzing the data, so that the database design, including the schema and the structure of data, could be arranged in a predefined and appropriate format, in order for the questions to be answered correctly.

Instead, in a *data driven* context, such as Big Data Analytics, we let the data *speak for themself:* without imposing a predefined schema to data, we can freely rearrange their format at real time, reflecting the correlations (hidden insights) that may arise, letting the analyst focus the attention on appropriate variables, as their relevance becomes evident as the flow of data from different sources are merged together to form a whole new domain of research.

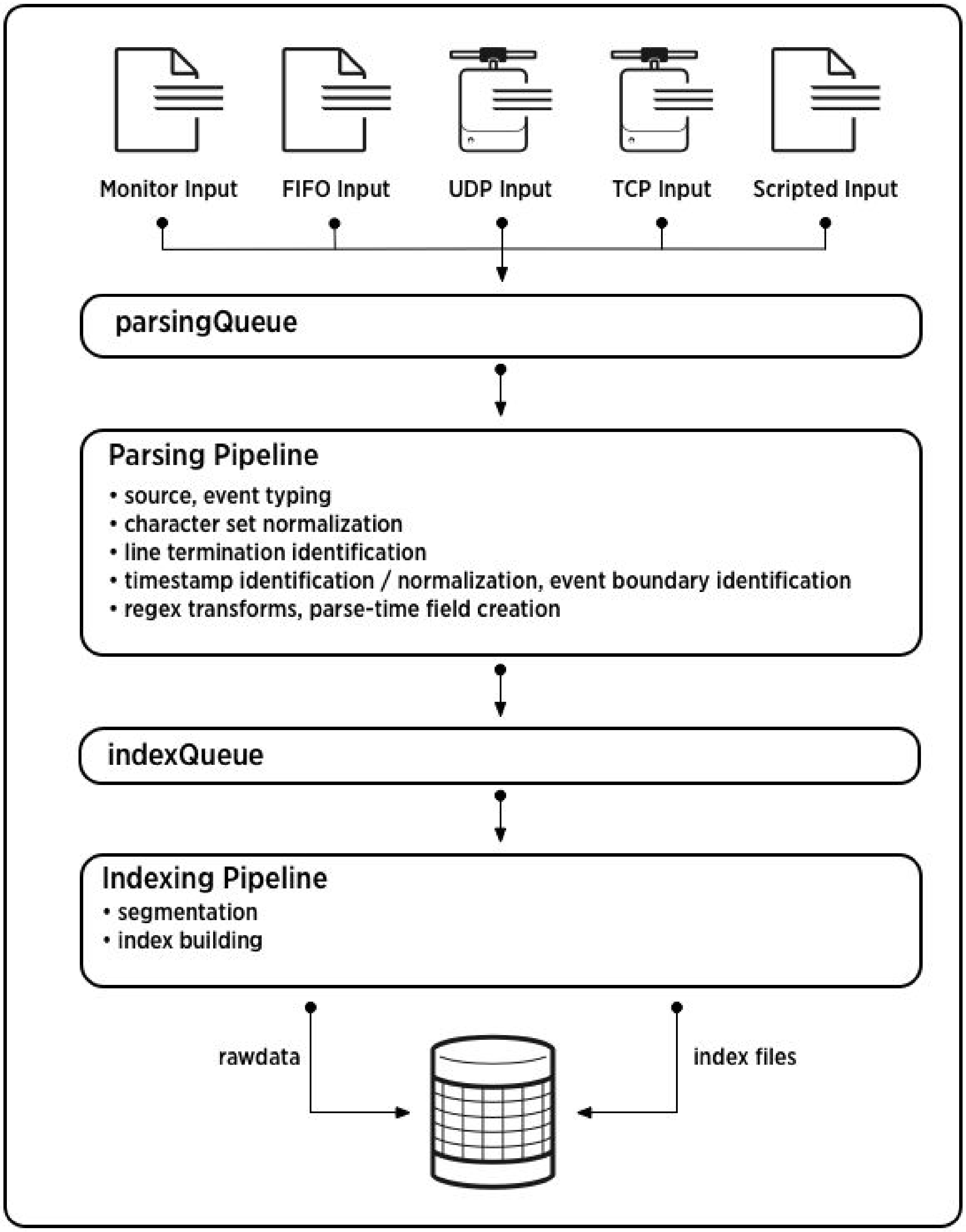
Chapter Two

What is Splunk?

As an event indexer, Splunk can perform *real-time data collecting, indexing and searching*, managing structured and unstructured textual data, both user-generated (i.e. Social media) and machine-generated (hardware appliances, sensors, etc): By handling the *Three Vs* very well, Splunk essentially offers three main (and autonomous scalable) functionalities:

- *Data collection*: The first step in using Splunk is to feed it with data. Basically, you point Splunk at data and it does the rest. In moments, you can start searching the data, or use it to create charts, reports, alerts, and other interesting outputs. Splunk can manage any kind of data; in particular, all IT streaming and historical data (such as event logs, web logs, live application logs, network feeds, system metrics, change monitoring, message queues, archive files, and so on). The data may reside on the same machine as the Splunk *indexer* (**local data**), or they can be located on another machine altogether (**remote data**).
- *Data indexing*: Once Splunk gets data, it immediately *indexes* them, so that they are available for searching. Splunk can *index* any type of *time-series data* (i.e. data with *timestamps*); leveraging on its universal indexing ability, Splunk transforms data into a series of individual *events*, consisting of searchable fields: this task is also known as **event processing**. *Event processing* occurs in two stages, **parsing** and **indexing**. During parsing, Splunk breaks data chunks into *events* which it hands off to the **indexing pipeline**, where final processing occurs (Figure

1).

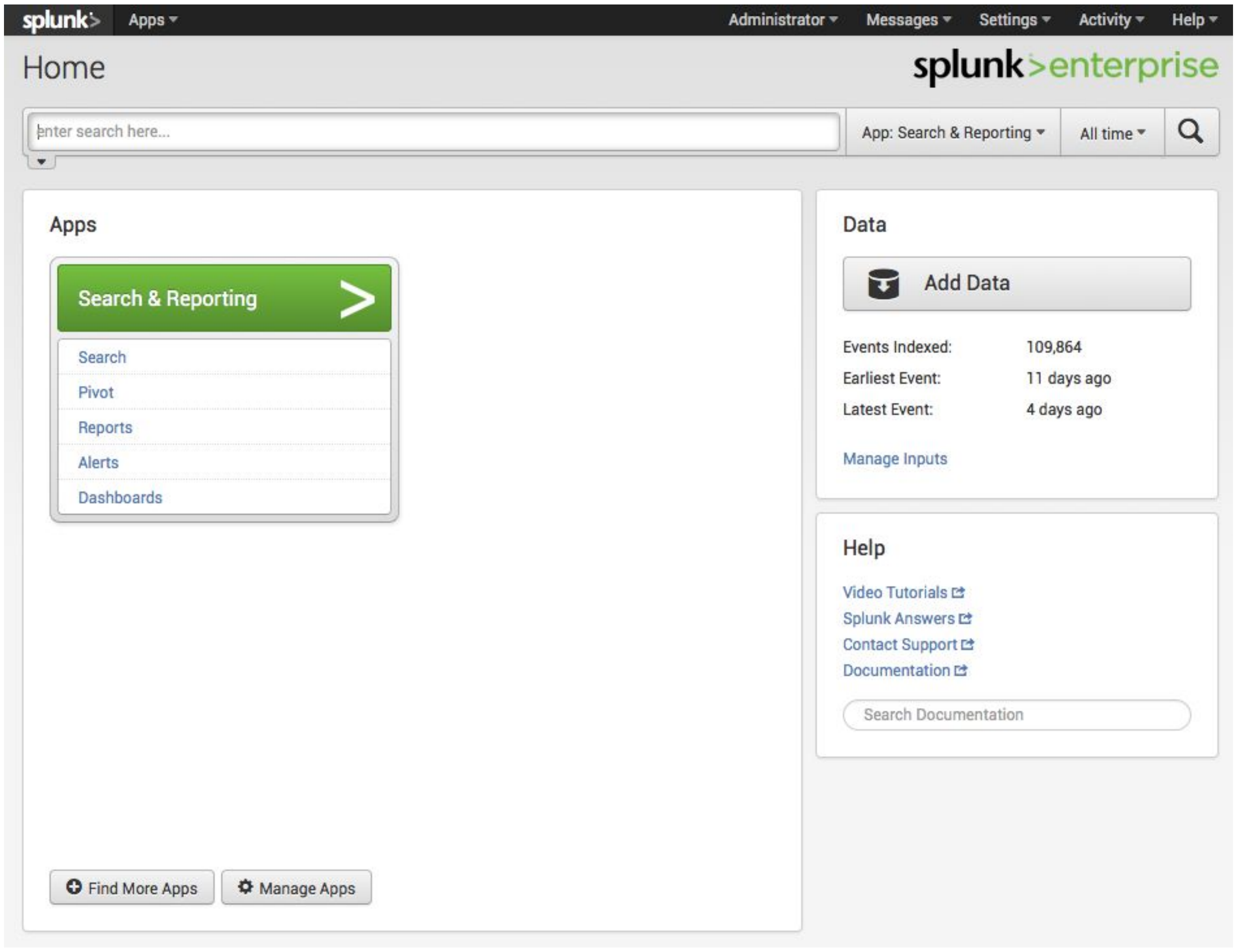

Monitor Input
FIFO Input
UDP Input
TCP Input
Scripted Input
parsingQueue
Parsing Pipeline
• source, event typing
• character set normalization
• line termination identification
• timestamp identification / normalization, event boundary identification
• regex transforms, parse-time field creation
indexQueue
Indexing Pipeline
• segmentation
• index building
rawdata
index files

- *Data Search*: once the data is indexed, you can begin searching it and manipulating the results using the **Splunk Search Language (SPL)**; the *search command* permits the use of keywords, phrases, fields, boolean expressions, along with comparison expressions, helping the analyst specifying exactly which events they want to retrieve from Splunk index(es).

Chapter Three

Splunk Search

To start searching, you need to point your browser to Splunk Home, i.e. the interactive portal to apps and data accessible from the Splunk instance (please note that Splunk Web interface is at http://localhost:8000), as in Figure 2:



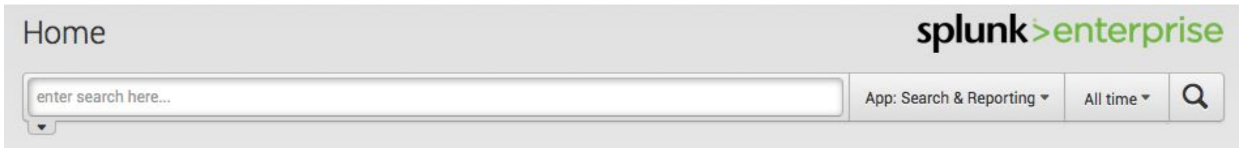
Events Indexed:	109,864
Earliest Event:	11 days ago
Latest Event:	4 days ago

(Figure 2)

In the Apps panel, you will see workspaces for the apps that are installed on your Splunk server that you have permission to view. The workspace displays a menu of the views and objects in the app context. Select the App to open it or select a content page listed in the workspace to go directly to that view. (Please note also the Data panel, that you can use to add data to Splunk).

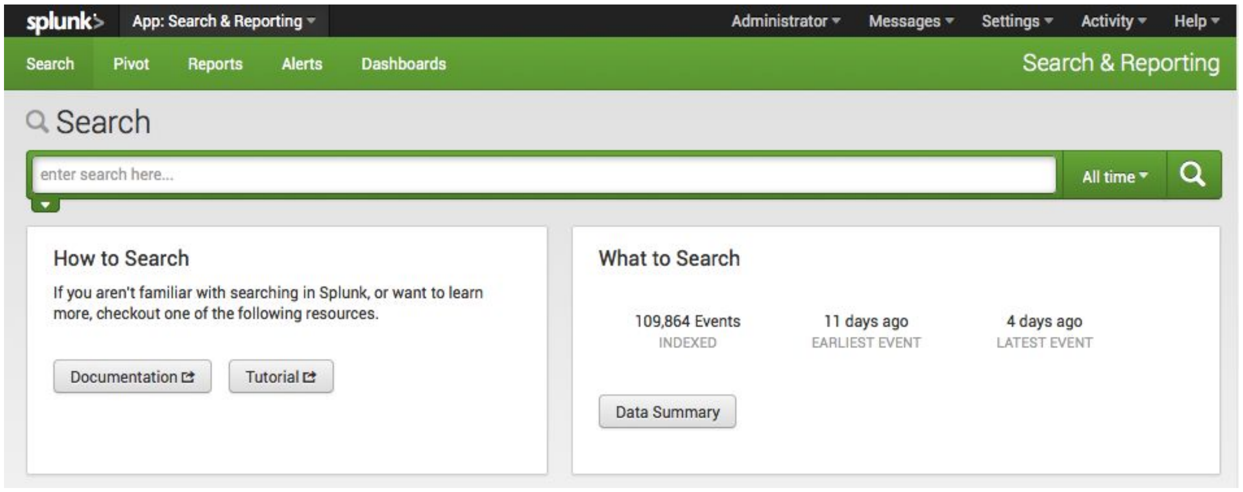
The app search bar

This *app search bar* is a shortcut that lets you run a search in a specified app context, without clicking through to the desired app. It is similar to the standard Splunk search bar and includes a time range picker. It also includes an App menu that allows you to select the app context in which to run your search (Figure 3)



(Figure 3)

The main parts of *Search* are the **search bar**, the **time range** picker, along with the ***How to search panel***, and the ***What to search*** panel (Figure 4):



If you aren't familiar with searching in Splunk, or want to learn more, checkout one of the following resources.

109,864 Events	11 days ago	4 days ago
INDEXED	EARLIEST EVENT	LATEST EVENT

(Figure 4)

The *search bar* is useful when running searches in Splunk Web: just type in the search string and hit enter or click the spyglass icon to the right of the time range picker.

The *time range picker* enables you to retrieve events over a specific time period: in case of *real-time searches* it is possible to specify a window over which to retrieve events; for *historical searches*, it is also possible to restrict search by specifying a relative time range or a specific date and time range, or selecting preset time ranges from the time range picker.

Chapter Four

Reports and Statistics tab

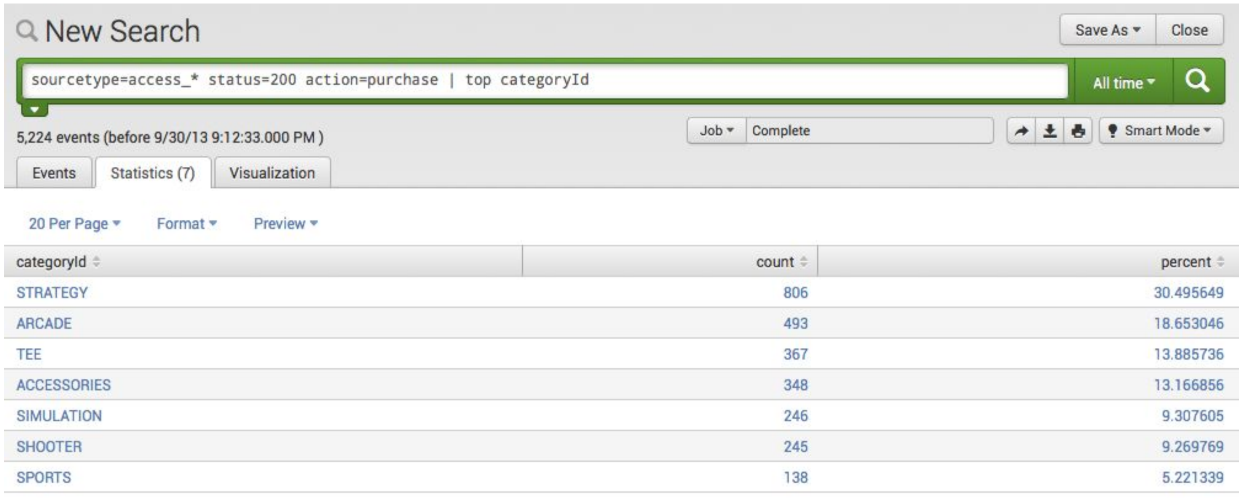
The results of a search are *reports*. In the example below (Figure 5) you can see the output of the following command search string:

sourcetype=access_* status=200 action=purchase | top categoryId

the command searches among all the web access types (*sourcetype=access_**) that resulted in a 200 HTTP access status (*status=200*), filtering the web requests that issued a purchase action (*action=purchase*) for a specific product, finally piping the output to the *top* command.

The top command returns a tabulated report for the most common values of *category_id* for the example of game product purchases (*STRATEGY, ARCADE*, *TEE*, etc).

Because top is a reporting command, this report displays in the *Statistics tab*.



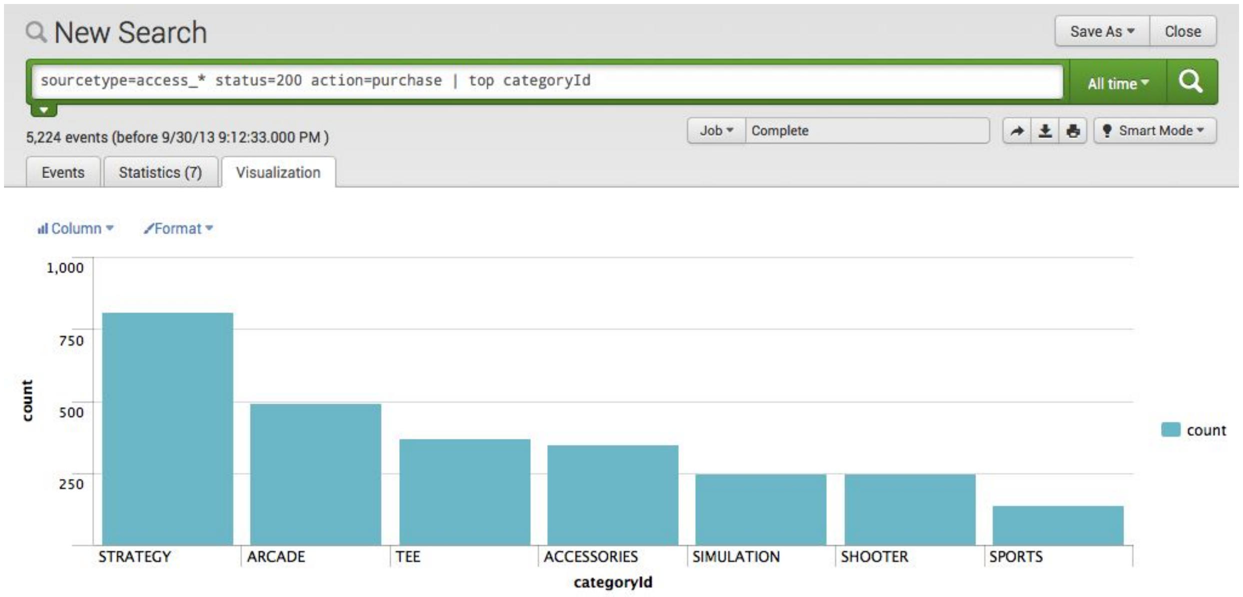
```
sourcetype=access_* status=200 action=purchase | top categoryId
```

5,224 events (before 9/30/13 9:12:33.000 PM)

categoryId	count	percent
STRATEGY	806	30.495649
ARCADE	493	18.653046
TEE	367	13.885736
ACCESSORIES	348	13.166856
SIMULATION	246	9.307605
SHOOTER	245	9.269769
SPORTS	138	5.221339

(Figure 5)

As top is a reporting command, you can also view (Figure 6) a report in the *Visualization tab*. By default, the Visualizations opens with a Column Chart, but of course you can choose the visualization that best fits your reporting needs (by clicking on the visualization type selector, recommended visualization charts for this data set, such as Column, Bar, and Pie charts, are shown).



```
sourcetype=access_* status=200 action=purchase | top categoryId
```

(Figure 6)

Chapter Five

Splunk Search Processing Language (SPL)

Splunk search functionality may be considered a real programming language, with its set of commands and functions that the user may implement in order to accomplish their research.

The search command syntax provides also the pipeline mechanism, in much the same fashion as unix-like command shell, so that consecutive commands are chained together using a pipe character, "|". The pipe character tells Splunk to use the output or result of one command (to the left of the pipe) as the input for the next command (to the right of the pipe).

Once you have set your *search command string*, you're ready to launch your search from Splunk Web in the Search app, or from the *command line interface (CLI)* or even making use of the Splunk REST API.

Splunk Search Command Examples

As a working example, we will use the *streamstats* command, that calculates summary statistics on search results; unlike *stats* (which works on the results as a whole), *streamstats* calculates statistics for each event at the time the event is seen. So it is quite useful whenever you need to diagnose attempts such user privilege escalation, or arp spoofing attacks, for example:

Example 1

Say that you'd like to track the cumulative count of distinct users that issue *download* event, provided that you track unique users on a daily basis; so the *search command string* would look like the following:

eventtype="download" | bin _time span=1d as day | stats values(clientip) as ips dc(clientip) by day | streamstats dc(ips) as "Cumulative total"

The *bin* command breaks the time into days, while the *stats* command calculates the distinct users (clientip) and user count per day; finally the *streamstats* command finds the running distinct count of users. This search returns a table that includes: day, ips, dc(clientip), and Cumulative total.

Example 2

This example uses *streamstats* to figure out when an IP address changes for a specific MAC address, say "AA:BB:CC:DD:00:00", in order for example to diagnose an *arp spoofing attack*: the *search command string* would resemble the following:

source=/Users/logs/arp.csv MAC= AA:BB:CC:DD:00:00 | head 10
| streamstats current=false last(IP_ADDRESS) as new_ip_addr last(_time) as time_of_change by MAC |
where IP_ADDRESS!=new_ip_addr | convert ctime(time_of_change) as time_of_change |
rename IP_ADDRESS as old_ip_addr | table time_of_change, MAC, old_ip_addr, new_ip_addr

The command outputs a table with fields *time_of_change, MAC, old_ip_addr, new_ip_addr*, so you can easily detect which ip address has been assigned to a specific MAC address, and at what time the change took place. (For more info about *streamstats* command options, please browse the command reference at: http://docs.splunk.com/Documentation/Splunk/6.0/SearchReference/streamstats).

Chapter Six

Splunk Approach in Security Threats Prevention

The real strength of Splunk may be seen in action when it comes to detecting complex threats in real time: in particular, in today's dynamic context, data often come in no particular structure and there is also the urgency to analyze it at the same time when events happen, rather than waiting for an extract, load, and translate process, as with traditional ETL procedures. By leveraging on different sources of data, and merging them to compound a wider picture of the events, it is possible to gain insights from correlations emerging by analyzing an extended domain of research. As an example of complex analysis in action, we will examine fraud detection, starting from the case of ATM withdrawals of funds, made by the *same customer* more than once in the last 15 minutes from *two different cities*:

sourcetype=ATM action=withdrawal | transaction customer maxspan=15m | eval location_count=mvcount(location) | where location_count>1 | stats values(location) by customer

In this search command string, we used the *transaction* command to group by customer in 15 minute spans, provided that we can reliably consider as fraud the case of customers withdrawing funds from two different locations.

As another example, we may consider the use case of *stolen credit cards*: in this use case, we may detect two different transactions of different imports issued at nearly the same time: the first transaction consisting in a purchase of small monetary amount, just to verify that the card is valid, and then a second purchase for a larger amount of money, carryed out as quick as possible to prevent suspicions.

sourcetype=card_transactions -earliest=15m | stats min(amount) as min max(amount) as max by customer | where min<50 AND max>500|table min, max, customer

In this example, we detect customer transactions for amounts less than 50 and greater than 500, carryed out in less than 15 minutes.

Conclusions

We met the power of Splunk in conducting real time complex analysis, based on different and heterogeneous sources of data; the only limit is the sky of analyst's imagination and creativity, in figuring out possible security tests to verify with Splunk support. Last but not least, Splunk is increasing its relevance as a general purpose Big Data platform, permitting the development of specific business apps, supporting organizations in their innovation management process.

Online Resources and References

- Splunk Documentation: http://docs.splunk.com/Documentation

Contacts

- Website: HackingWisdom
- Twitter: @HackingWisdom