CA : 3

Name – Mukesh Yadav

Reg no. 11701592

Subject – INT301

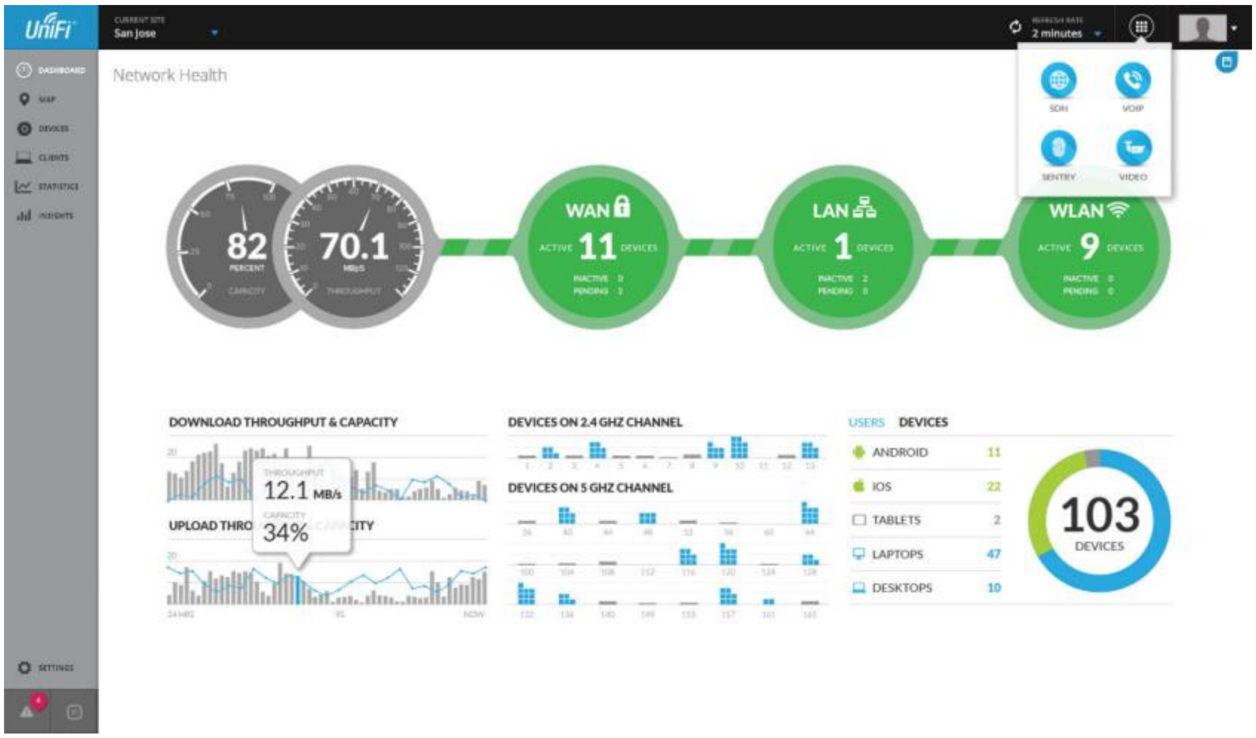
Section – KE057

Roll no : 01

Submitted To

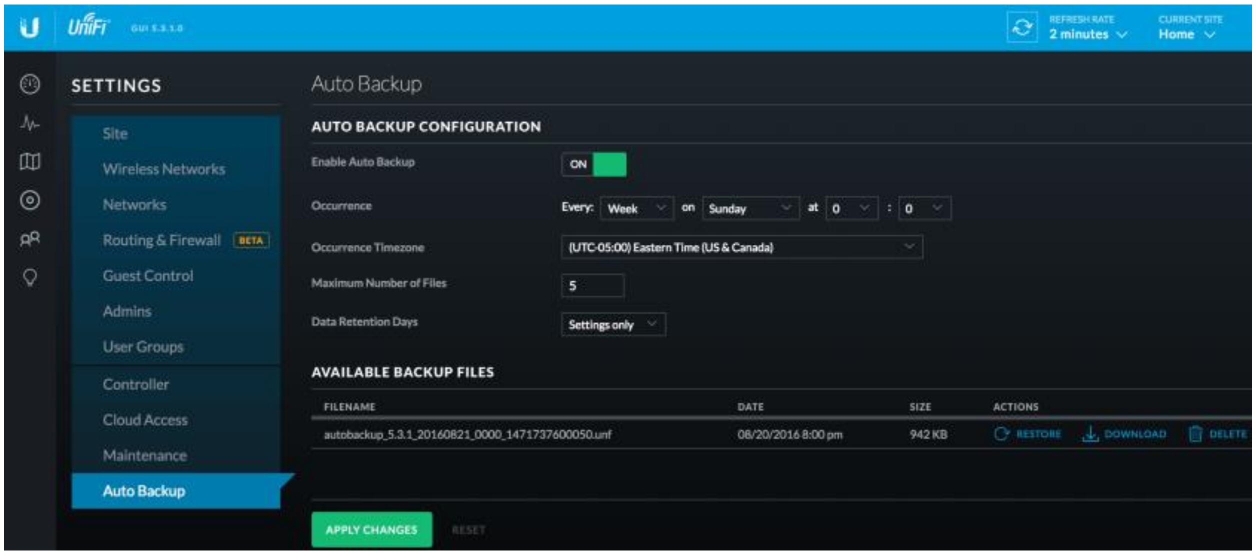
Navjot Kaur

this time I want to partake one of my favored side conditioning playing with my Ubiquiti home setup! As you formerly know I've my regulator running on Azure, but I wanted to understand more on which kind of data is stored inside the regulator, in other words where the data that we see in the regulator dashboard is stored.



Examining the binaries and looking in 2- 3 posts on the forums I figured out that this data is sitting in a mongodb database, but I wanted of course to look a bit inside of this database.

What I did is the following, I made a backup of the data of the regulator using the web interface of the regulator and I downloaded it locally.



Auto Backup

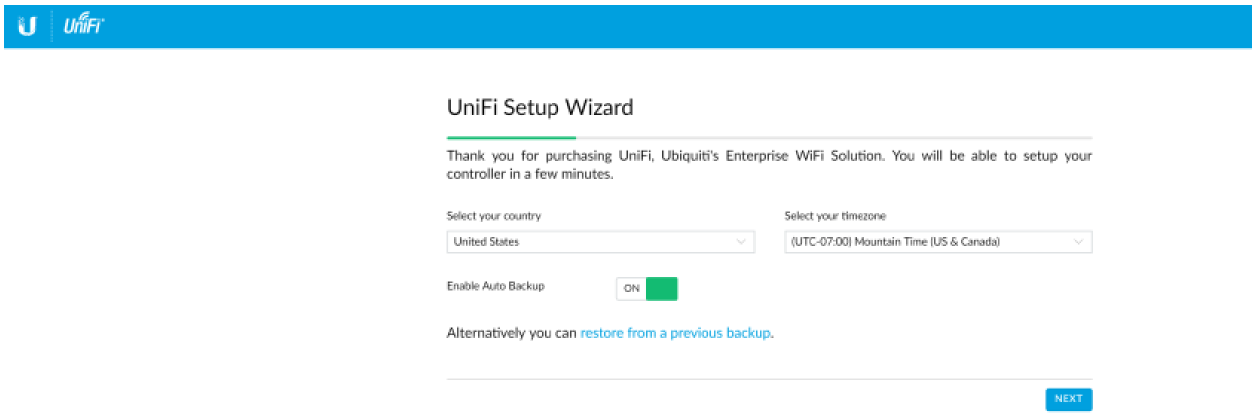
AUTO BACKUP CONFIGURATION

Enable Auto Backup	ON
Occurrence	Every: Week on Sunday at 0 : 0
Occurrence Timezone	(UTC-05:00) Eastern Time (US & Canada)
Maximum Number of Files	5
Data Retention Days	Settings only

AVAILABLE BACKUP FILES

FILENAME	DATE	SIZE	ACTIONS
autobackup_5.3.1_20160821_0000_1471737600050.unf	08/20/2016 8:00 pm	942 KB	RESTORE DOWNLOAD DELETE

At this stage I downloaded the regulator software for an installation on my laptop(Window) and at regulator incipiency i requested a restore of the backup i just downloaded from the pall regulator.



UniFi Setup Wizard

Thank you for purchasing UniFi, Ubiquiti's Enterprise WiFi Solution. You will be able to setup your controller in a few minutes.

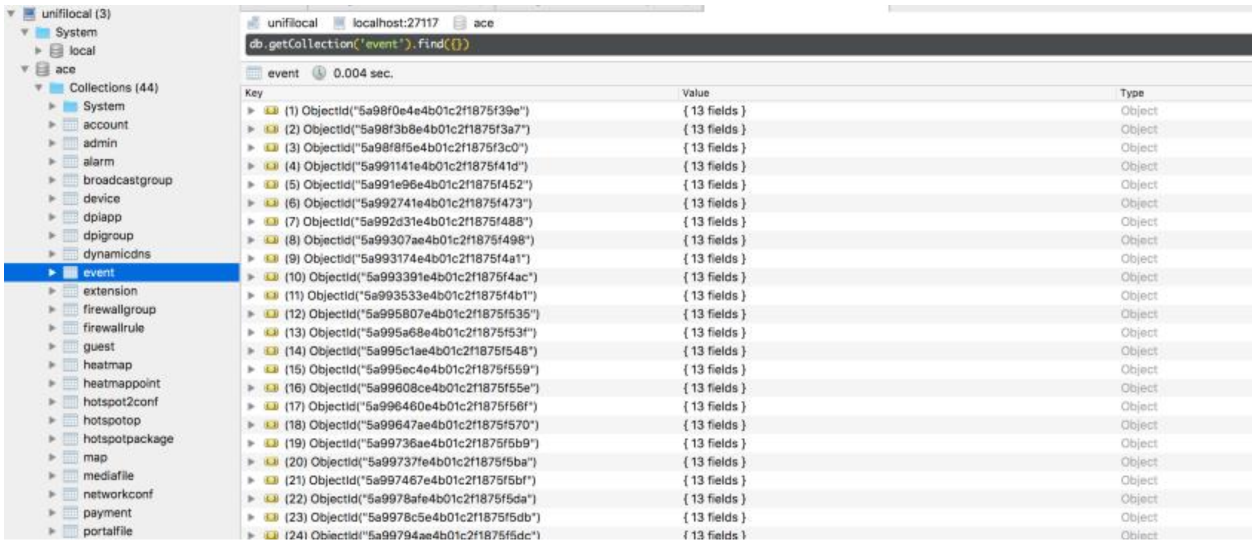
Select your country: United States

Select your timezone: (UTC-07:00) Mountain Time (US & Canada)

Enable Auto Backup: ON

Alternatively you can restore from a previous backup.

Once the restore is done, mantain the regulator running and you can use a mongodb customer like robo3T and connect to localhost and harborage 27117(we connect to the mongod process started by the regulator locally).

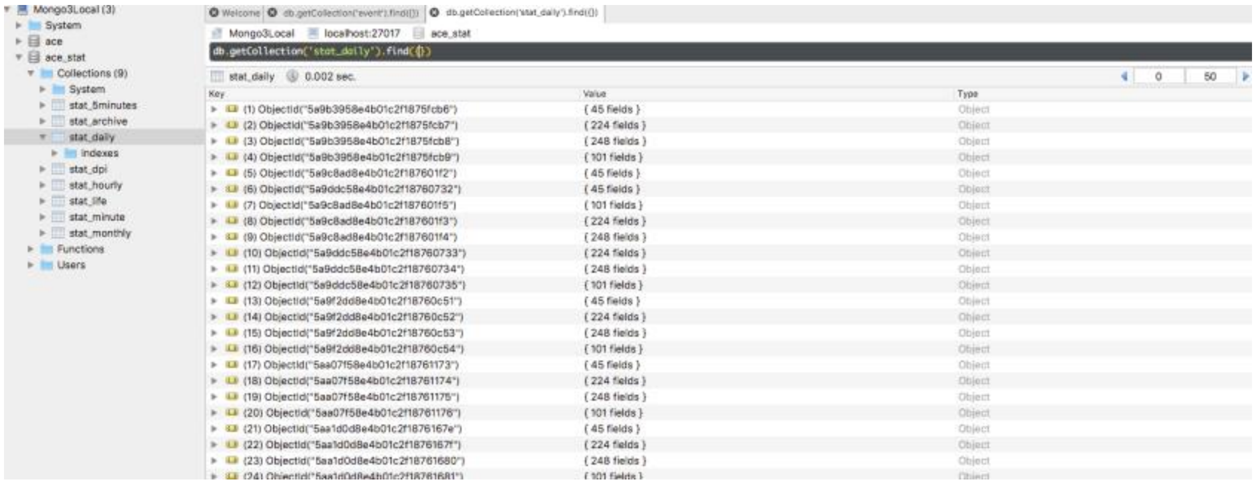


```
unifilocal (3)
  System
    local
  ace
    Collections (44)
      System
      account
      admin
      alarm
      broadcastgroup
      device
      dpiapp
      dpigroup
      dynamicdns
      event
      extension
      firewallgroup
      firewallrule
      guest
      heatmap
      heatmappoint
      hotspot2conf
      hotspotop
      hotspotpackage
      map
      mediafile
      networkconf
      payment
      portalfile
```

```
unifilocal   localhost:27117   ace
db.getCollection('event').find({})

event   0.004 sec.
Key                                              Value            Type
(1) ObjectId("5a98f0e4e4b01c2f1875f39e")         { 13 fields }    Object
(2) ObjectId("5a98f3b8e4b01c2f1875f3a7")         { 13 fields }    Object
(3) ObjectId("5a98f8f5e4b01c2f1875f3c0")         { 13 fields }    Object
(4) ObjectId("5a991141e4b01c2f1875f41d")         { 13 fields }    Object
(5) ObjectId("5a991e96e4b01c2f1875f452")         { 13 fields }    Object
(6) ObjectId("5a992741e4b01c2f1875f473")         { 13 fields }    Object
(7) ObjectId("5a992d31e4b01c2f1875f488")         { 13 fields }    Object
(8) ObjectId("5a99307ae4b01c2f1875f498")         { 13 fields }    Object
(9) ObjectId("5a993174e4b01c2f1875f4a1")         { 13 fields }    Object
(10) ObjectId("5a993391e4b01c2f1875f4ac")        { 13 fields }    Object
(11) ObjectId("5a993533e4b01c2f1875f4b1")        { 13 fields }    Object
(12) ObjectId("5a995807e4b01c2f1875f535")        { 13 fields }    Object
(13) ObjectId("5a995a68e4b01c2f1875f53f")        { 13 fields }    Object
(14) ObjectId("5a995c1ae4b01c2f1875f548")        { 13 fields }    Object
(15) ObjectId("5a995ec4e4b01c2f1875f559")        { 13 fields }    Object
(16) ObjectId("5a99608ce4b01c2f1875f55e")        { 13 fields }    Object
(17) ObjectId("5a996460e4b01c2f1875f56f")        { 13 fields }    Object
(18) ObjectId("5a99647ae4b01c2f1875f570")        { 13 fields }    Object
(19) ObjectId("5a99736ae4b01c2f1875f5b9")        { 13 fields }    Object
(20) ObjectId("5a99737fe4b01c2f1875f5ba")        { 13 fields }    Object
(21) ObjectId("5a997467e4b01c2f1875f5bf")        { 13 fields }    Object
(22) ObjectId("5a9978afe4b01c2f1875f5da")        { 13 fields }    Object
(23) ObjectId("5a9978c5e4b01c2f1875f5db")        { 13 fields }    Object
(24) ObjectId("5a99794ae4b01c2f1875f5dc")        { 13 fields }    Object
```

I would like to produce some nice dashboards, with some visualization tool like Tableau or Power Bi or simply Excel but the data in a " Document " format while I need it to be in Table/ Records format. The result is the Mongo Bi Connector that is a kind of " wrapper " or " translator " between the " Document " world and the tables records world. But effects are no way simple;-), this connector works only from MongoDBv.3.0 or advanced while the one inside the regulator software is2.6. So first we've to download a separate mongodb garçon that works with it but more importantly upgrade the database itself to the3.X format. First stop's dupe the database from the regulator brochure(check a brochure called db) and copy it to another position, write down this position. I tested and failed colorful times before understanding how to do it but this is the sequence(using pop to install mongodb on my mackintosh) install mongodb3.0 –> open the regulator database in the position we copied. uninstall3.0/ install3.2 –> open the regulator database in the position we copied. uninstall3.2 install3.4 –> open the regulator database in the position we copied. This will bring the database to a format that's working with the Bi Connector.



- Mongo3Local (3)
  - System
  - ace
  - ace_stat
    - Collections (9)
      - System
      - stat_5minutes
      - stat_archive
      - stat_daily
        - indexes
      - stat_dpi
      - stat_hourly
      - stat_life
      - stat_minute
      - stat_monthly
    - Functions
    - Users

Welcome | db.getCollection('event').find({}) | db.getCollection('stat_daily').find({})

Mongo3Local localhost:27017 ace_stat

```
db.getCollection('stat_daily').find({})
```

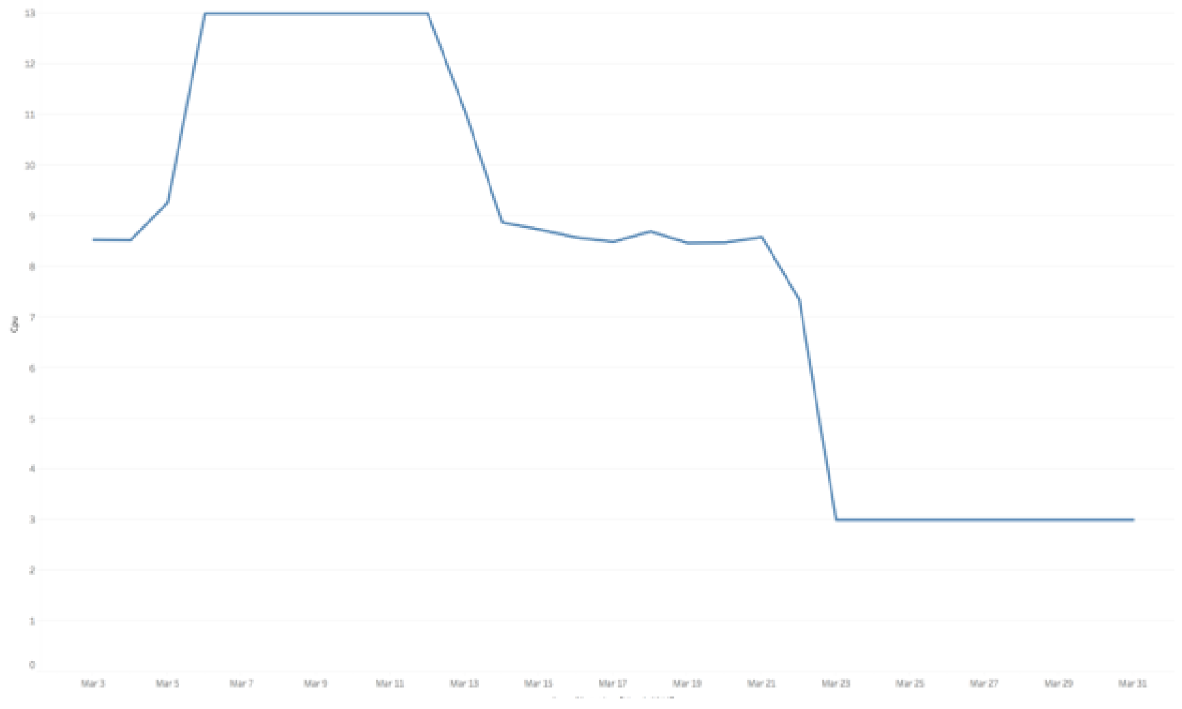
stat_daily 0.002 sec. 0 50

Key	Value	Type
(1) ObjectId("5a9b3958e4b01c2f1875fcb6")	{ 45 fields }	Object
(2) ObjectId("5a9b3958e4b01c2f1875fcb7")	{ 224 fields }	Object
(3) ObjectId("5a9b3958e4b01c2f1875fcb8")	{ 248 fields }	Object
(4) ObjectId("5a9b3958e4b01c2f1875fcb9")	{ 101 fields }	Object
(5) ObjectId("5a9c8ad8e4b01c2f187601f2")	{ 45 fields }	Object
(6) ObjectId("5a9ddc58e4b01c2f18760732")	{ 45 fields }	Object
(7) ObjectId("5a9c8ad8e4b01c2f187601f5")	{ 101 fields }	Object
(8) ObjectId("5a9c8ad8e4b01c2f187601f3")	{ 224 fields }	Object
(9) ObjectId("5a9c8ad8e4b01c2f187601f4")	{ 248 fields }	Object
(10) ObjectId("5a9ddc58e4b01c2f18760733")	{ 224 fields }	Object
(11) ObjectId("5a9ddc58e4b01c2f18760734")	{ 248 fields }	Object
(12) ObjectId("5a9ddc58e4b01c2f18760735")	{ 101 fields }	Object
(13) ObjectId("5a9f2dd8e4b01c2f18760c51")	{ 45 fields }	Object
(14) ObjectId("5a9f2dd8e4b01c2f18760c52")	{ 224 fields }	Object
(15) ObjectId("5a9f2dd8e4b01c2f18760c53")	{ 248 fields }	Object
(16) ObjectId("5a9f2dd8e4b01c2f18760c54")	{ 101 fields }	Object
(17) ObjectId("5aa07f58e4b01c2f18761173")	{ 45 fields }	Object
(18) ObjectId("5aa07f58e4b01c2f18761174")	{ 224 fields }	Object
(19) ObjectId("5aa07f58e4b01c2f18761175")	{ 248 fields }	Object
(20) ObjectId("5aa07f58e4b01c2f18761176")	{ 101 fields }	Object
(21) ObjectId("5aa1d0d8e4b01c2f1876167e")	{ 45 fields }	Object
(22) ObjectId("5aa1d0d8e4b01c2f1876167f")	{ 224 fields }	Object
(23) ObjectId("5aa1d0d8e4b01c2f18761680")	{ 248 fields }	Object
(24) ObjectId("5aa1d0d8e4b01c2f18761681")	{ 101 fields }	Object

Now in the Bi Connector you can prize the schema of a document collection you like(for illustration the stat daily collection of ace stat database) and after that generate the wrapper process that can be used by a visualization tool

```
2023-04-02T15:58:53.021+0200 I SAMPLER [schemaDiscovery mapped schema for
51 namespaces: "ace" (43): ["hotspotpackage","broadcastgroup","firewa
llgroup", "radiusprofile","heatmappoint", "hotspot2conf", "firewallrule
", "scheduletask","verification","portforward","networkconf","port
alfile","rogueknown","dynam icdns","wlangroup","usergroup","hotspotop","
privilege","extension", "mediafile","portconf","dpigroup","wlanconf","rout
ing","setting","account","vouc1er","payment", "heatmap","dpiapp", "d
evice", "alarm","guest","event","rogue", "admin","site", "task","wall"
,"stat","user", "map","tag"]; "ace_stat" ( 3): ["stat_5mintes"
,"stat_archive","stat_monthly","stat_minute","stat hour!'l","stat_daily"
,"stat_life","stat_dpi"]
```

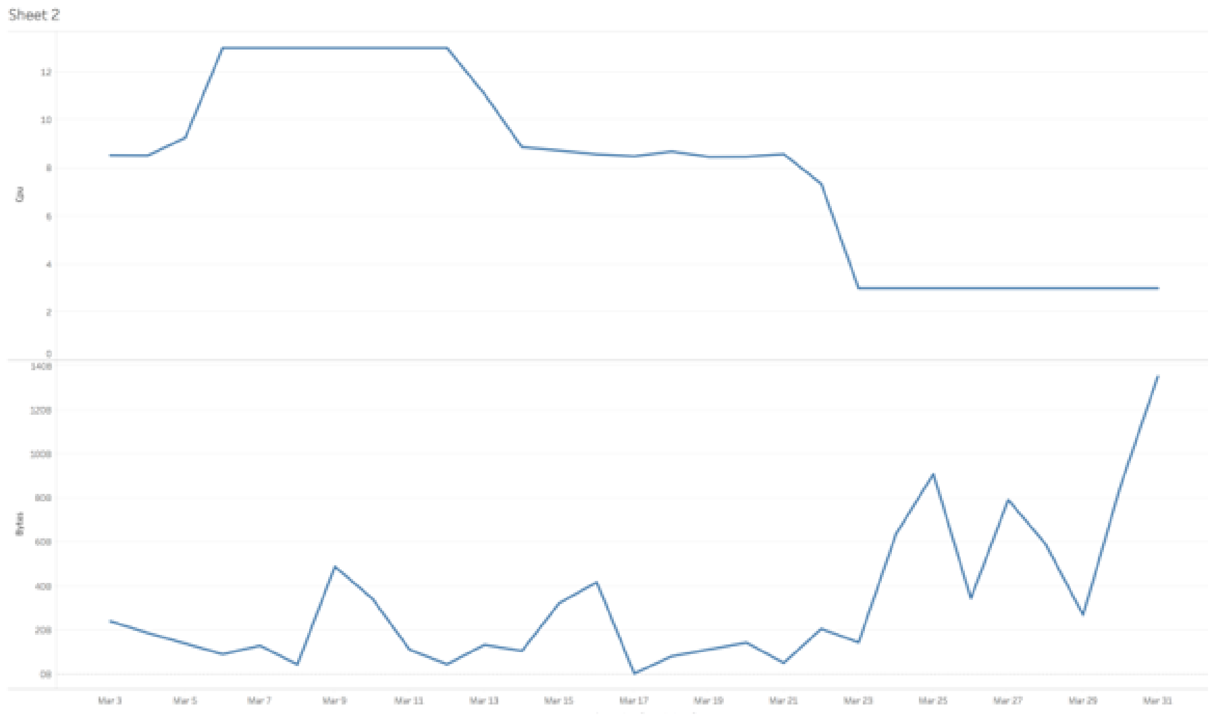
In my case I used tableau to produce some test dashboards.



a Day of Datetime [march 2023]

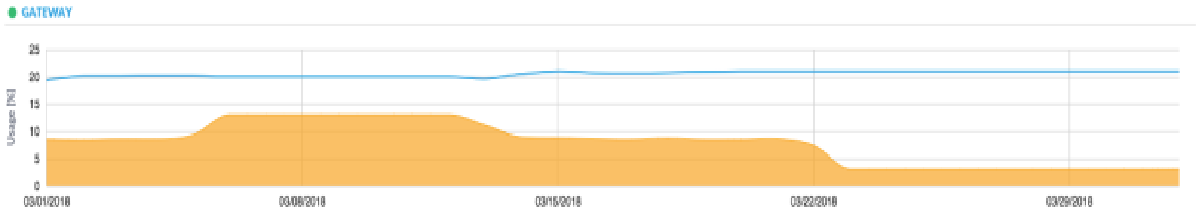
Then I see that the CPU of my gateway was a bit high during the first part of the month and also dropped significantly.

I can add other criteria like downloaded data, etc. to understand better.



b Day of Datetime [march 2023]

In reality in this specific case there's formerly a super nice visualization formerly offered by the regulator dashboards.



So the real intriguing thing then is that you can actually produce your own report and also discover new perceptivity looking at the your own network data

This is custom reports on your Unifi network and device data!

Report: Analyzing Malicious Files using Open Source Software

Introduction

1.1 Objective of the Project

The objective of this project is to analyze malicious files using open source software. The project aims to provide an efficient and effective solution for detecting and analyzing malware in a system. By utilizing open source software, the project aims to reduce the complexity and cost of malware analysis.

1.2 Description of the Project

The project involves analyzing malicious files by tracing API calls and behavior of files using open source software. The software is used to monitor the behavior of the file and the system calls it makes during its execution. The collected information is then analyzed to detect the presence of malware. In addition, the project includes the implementation of automatic upgrade/update of UniFi Network Controller / Network Application.

1.3 Scope of the Project

The scope of the project is to analyze various types of malicious files and detect their presence in a system. The project aims to provide an effective solution to detect and analyze malware in a system. The project also includes the implementation of automatic upgrade/update of UniFi Network Controller / Network Application to ensure the system remains secure.

System Description

2.1 Target System Description

The target system for the project is a computer system running on a Windows operating system. The system should have open source software installed for monitoring and analyzing the behavior of files and API calls. In addition, the UniFi Network Controller / Network Application should be installed for automatic upgrade/update.

2.2 Assumptions and Dependencies

The project assumes that the system is running on a Windows operating system and has the required open source software installed. In addition, the project depends on the availability of a malware dataset for testing and analysis.

2.3 Functional/Non-Functional Dependencies

The functional dependencies of the project include the availability of open source software for tracing API calls and behavior of files. The project also depends on the availability of a malware dataset for testing and analysis. The non-functional dependencies of the project include the system performance and the accuracy of the malware detection and analysis.

2.4 Dataset used in support of the project

The project uses a malware dataset for testing and analysis. The dataset includes various types of malware, such as viruses, Trojans, and worms. The dataset is used to test the effectiveness and accuracy of the malware detection and analysis.

Analysis Report

The analysis of malicious files using open source software involves monitoring the behavior of files and API calls. The collected data is then analyzed to detect the presence of malware. The following steps are involved in analyzing malicious files using open source software:

Step 1: Collecting Data

The first step in analyzing malicious files is to collect data on the behavior of the file and the system calls it makes during its execution. Open source software such as Process Monitor and API Monitor can be used to collect this data.

Step 2: Analyzing Data

Once the data has been collected, it is analyzed to detect the presence of malware. This involves looking for suspicious behavior such as attempts to modify system files, create new files, or connect to suspicious IP addresses.

Step 3: Reporting

The results of the analysis are reported, and the presence of malware is confirmed or ruled out. If malware is detected, the type of malware is identified, and the appropriate action is taken to remove it.

Step 4: Conclusion

In conclusion, analyzing malicious files using open source software is an effective and efficient solution for detecting and analyzing malware in a system. The project aims to reduce the complexity and cost of malware analysis by utilizing open source software. By implementing automatic upgrade/update of UniFi Network Controller / Network Application, the project ensures that the system remains secure.

https://github.com/mdn/content/edit/main/files/en-us/glossary/copyleft/index.md