truth Documentation

Release 0.1

truthadjustr

Oct 26, 2017

Contents

Welcome.

CHAPTER 1

ANALYTICS

graylog

A Java server program that uses MongoDB and ElasticSearch to collect and later on do query of accumulated and collected logs.

Traditionally, data analytics uses these three technologies together commonly known as E.L.K:

- ElasticSearch (for data querying using Lucene syntax)
- Logstash (for data pushing into ElasticSearch?)
- Kibana (for data visualization)

graylog only uses ElasticSearch.

CHAPTER 2

UNIX

The first one creates a **self-signed** certificate, while the second creates a **CSR**:

```
openssl req -sha256 -new -x509 -key app.key.pem -out app.cert.pem  # creates self-
↪signed cert
openssl req -sha256 -new -key app.key.pem -out app.csr.pem         # creates a CSR
```

A *certificate* has already a date validity and expiration tagged into the certificate whereas a *CSR* does not have any concept of date validity and expiration yet.

A xxxx.p12 file is a cryptographic container file that can be password-protected. This file contains both the private & public keys and including the supporting certificate chain of trusts.. To analyse this file,:

```
openssl pkcs12 -in xxxx.p12 -out xxxx.cert.pem -clcerts -nokeys   # <--- retrieve the␣
↪certificate (and publick key)
openssl pkcs12 -in xxxx.p12 -out xxxx.privkey.pem -nocerts -nodes # <--- retrieve the␣
↪private key
```

The CRL can be downloaded from the certificate if you can see a CRL URL inside. But the CRL is in DER format. Convert to pem first:

```
openssl crl -inform DER -in crl.der -outform PEM -out crl.pem
```

And then read inside whose serial is expired:

```
openssl crl -in crl.pem -noout -text
```

Get the additional certificate chain inside .p12:

```
openssl pkcs12 -in path.p12 -out newfile.crt.pem -nokeys
```

AIX

HMC Tutorial:

- https://www.youtube.com/watch?v=FWX3fWSfAIA

Linux

CHAPTER 3

SHELLS

Bourne Shell

Korn Shell

Bourne Again Shell

CHAPTER 4

SCRIPTING

Python

Ruby

I highlight below an aspect of a programming language that just assumes too much and is hard to make sense of.

```
require 'open-uri'
require 'redis'

URLS = %w[
    http://www.gutenberg.org/ebooks/98.txt.utf-8
    http://www.gutenberg.org/ebooks/1400.txt.utf-8
    http://www.gutenberg.org/ebooks/730.txt.utf-8
    http://www.gutenberg.org/ebooks/766.txt.utf-8
    http://www.gutenberg.org/ebooks/19337.txt.utf-8
    http://www.gutenberg.org/ebooks/700.txt.utf-8
]

BOOKS = URLS.map(&File.method(:basename))
REDIS = Redis.current

URLS.each do |url|
    text = open(url)
    name = File.basename(url)

    text.each_line do |line|
        REDIS.pfadd(name,line.split(/\s+/).map(&:downcase))
    end
end

BOOKS.each do |name|
    puts "#{name}: #{REDIS.pfcount(name)}"
end
```

```
puts "All: #{REDIS.pfcount(*BOOKS)}"
```

Lua

Perl

CHAPTER 5

PROGRAMMING

C

Java

Go

CHAPTER 6

IOT

The Internet of things.

CHAPTER 7

REDIS

Client stunnel's config:

```
key = /tmp/stunnel/client.key.pem
cert = /tmp/stunnel/client.cert.pem
CAfile = /tmp/stunnel/chain-of-trust.pem
verify = 2
client = yes
pid = /tmp/stunnel/stunnel.pid
fips = no
[redis]
accept = 127.0.0.1:6379
connect = 172.17.0.11:9379
```

Server stunnel's config:

```
pid = /var/run/stunnel.pid
verify = 2
CAfile = /etc/stunnel/chain-of-trust.pem

[redis]
accept = 9379
connect = 127.0.0.1:6379
cert = /etc/stunnel/webdis.cert.pem
key = /etc/stunnel/webdis.key.pem
```

CHAPTER 8

DOCKER

It is all about:

- chroot
- namespaces
- control groups

CHAPTER 9

Indices and tables

- genindex
- modindex
- search