**#tinkoff fintech**

homework #8

MONITORING,   
LIVE RESPONSE, **THREAT HUNTING**

**30.04.2019**

# Tasks#:

### Установить стек ELK (Есть докер контейнеры)

### Установить на мониторинг виртуальную машину Windows 7, Flare VM.

### УСТАНОВИТЬ SYSMON

### УСТАНОВИТЬ WINLOGBEAT

### ПЕРЕНАПРАВИТЬ СОБЫТИЯ ЖУРНАЛОВ SYSMON И SECURITY В ELK

### Настроить дашборды или фильтры для детектирования техники атаки Credential Dumping

#### 

# ПРЕДВАРИТЕЛЬНЫЙ ЭТАП

**Небольшое определение своими словами и общее понимание что предстоит ниже напишу на этой страницу. А также расскажу примерный план, что как и с чем должно взаимодействовать, основные настройки сети и прочие вступительные этапы.**

**ELK STACK aka Elastic Stack** - *по сути объединение нескольких open source проектов в первую очередь для агрегации данных (в нашем случае логов), их фильтрации, анализа и визуализации.*

### Очень базово (больше для себя) распишу:

**Elasticsearch** - для поиска и анализа

**Logstash** - своеобразный фильтр-агрегатор-парсер логов из различных источников

**Beats** (Filebeats, Winlogbeats) - лог-шипперы (по факту - узконаправленные приложения предоставляющие логи либо напрямую elasticsearch, либо в логстэш.

**Kibana** - визуализатор (и не только) данных.

**Nginx** - web-server

С определениями разобрались. **Основная задача настроить сборщик логов на системе Windows 7, в частности через Sysmon. И среди полученных данных настроить фильтр на событие типа Credential Dumping.** В качестве финального результата можно настроить Dashboard.

Для себя, решил дополнительно организовать небольшую сеть виртуальных машин, чтобы оценить Elk Stack чуть больше и глубже :)

### СОБСТВЕННО, КАК И С ЧЕМ Я ПЛАНИРУЮ ВЫПОЛНИТЬ УСТАНОВКУ СТЕКА.

**Хост-машина (по сути а-ля командный центр):**

Arch-based Linux

**Виртуальный сервер с ELK Stack:**

Ubuntu LTS 18.04 Server headless

**Виртуальная машина для атаки Credential Dumping:**

Windows 7 (Flare edition)

**Среда виртуализации:**

Virtualbox

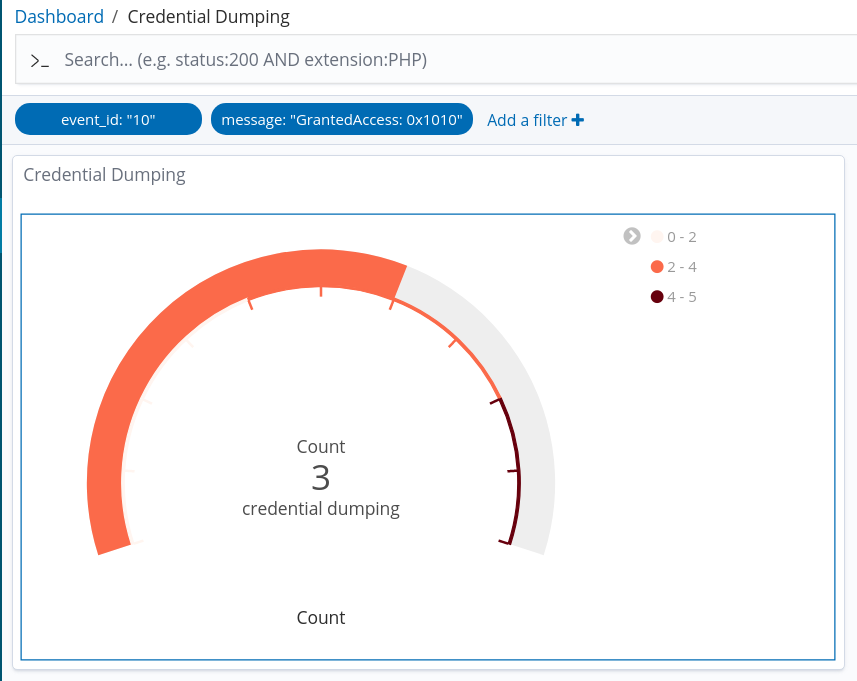
**Организация сети:**

Virtual Host-Only network (192.168.56.1 / 24)

**Почему не docker?**

Было время установить и сконфигурировать всё самому. К тому же, стек понравился, захотел узнать его получше, изнутри тоже :)

**Что должно быть в финале этой лабораторной?**



#### УСТАНОВКА СТЕКА

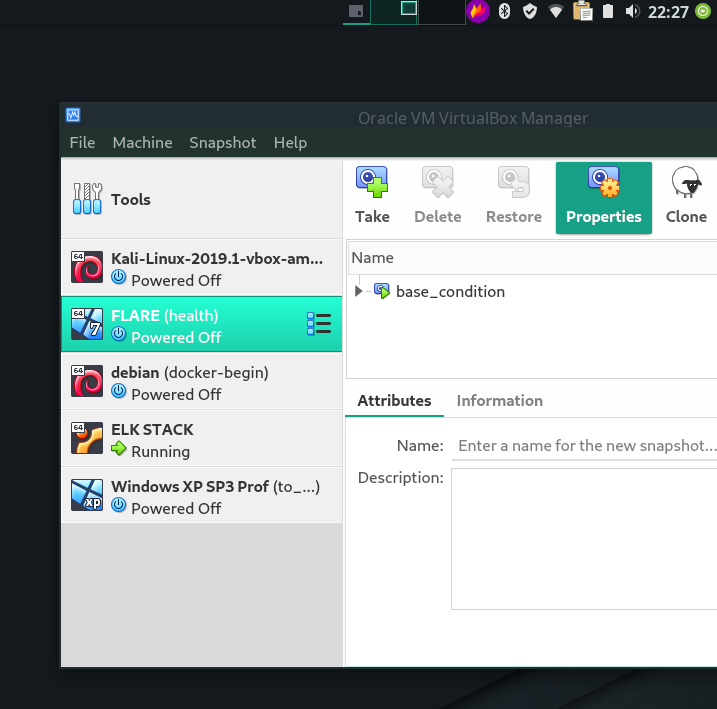
## В общем, чуть ниже наделаю скриншоты своих виртуалок, сетевых адаптеров, каких-то моментов в настройке. А так, общее впечатление позитивное. Стек установился конечно не сразу, так как не все вещи были очевидны, переизбыток конфиг файлов тоже сказывался, + стек новый, пришлось повникать.

## 

## Какие-то ключевые моменты в установке, это пожалуй**, поставить правильную java, согласовать все версии приложений в стеке** (чтобы не получилось как у меня, kibana 7.0, а какой-нибудь filebeat 6.7.1), **внимательно править конфиги** исходя из настроек своей сети, **делать всё поэтапно,** проверяя каждый этап хотя бы элементарными systemctl status <name>, ну и **man google** ;)

## 

## Все команды не буду описывать и скринить, но ключевые моменты отражу ниже:



Собственно всё что есть в Virtualbox

## Запускаю сервер с установленным elk-stack (ubuntu):

vboxmanage startvm "ELK STACK" --type headless

## Затем предварительно настроив ssh коннектимся на на наш сервер.

ssh root@192.168.56.106 -p 22

## 

## Процесс установки стэка прямолинейный. Ставим все необходимые для развертки стэка приложения попутно правя конфиги.

## Типичный конфиг файл <filename>.yml. В каждом из них прописываем адрес нашего сервера (192.68.56.106) а также необходимые порты.

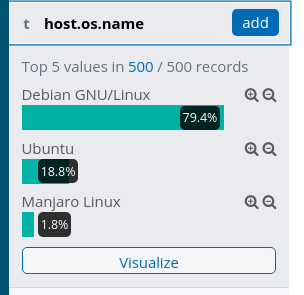
## 

Немного сложнее настройка логстэша. Так как мы не отправляем напрямую логи с filebeats / winlogbeats в elasticsearch нам надо сконфигугрировать все сервисы так, чтобы логи шли через логстэш. А для него самого тоже нужны определенные кофиг файлы. На скрине ниже видно что у меня есть отдельно конфиг для INPUT , FILTER и OUTPUT. Собственно, в этом вся прелесть logstash, возможность парсить логи с учетом нужных фильтров для дальнейшего их вывода в кибану. В нашем случае тут примитивная настройка для syslog логов.

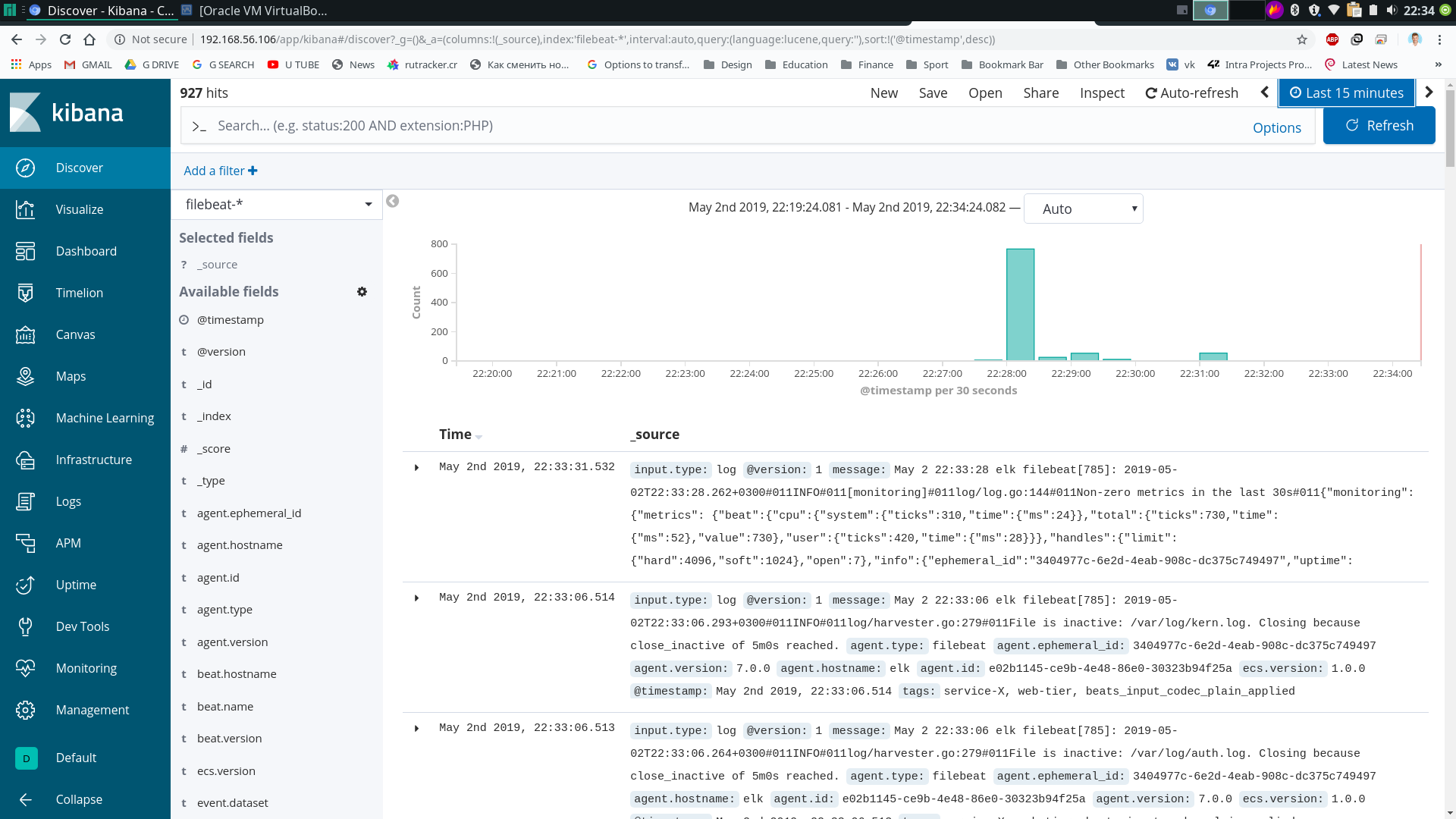
## 

## 

## В кибане очень удобно фильтровать логи от filebeat на разных машинах с помощью фильтров.



## 



## Думаю, что теперь можно посмотреть нашу Windows машину, и настроить на ней, для начала Winlogbeat service.

#### 2. УСТАНОВКА WNLOGBEAT (SYSMON)

## Вообще, есть отличный мануал на офф сайте элка по установке winlogbeat на винду, поэтому перепечатывать step-by-step гайд не буду. Там все типично, скачиваем архив, распаковываем. Устанавливаем службу (не забыв про Set-ExecutionPolicy UnRestricted). Но пока службу не запускаем, так как надо сконфигурировать файл winlogbeat.yml

## 

## 

## Вот тут есть одна строчка, объяснение которой я дам ниже, когда буду описывать установку sysmon. По сути, на этом скрине я объявляю откуда пойдут event логи.

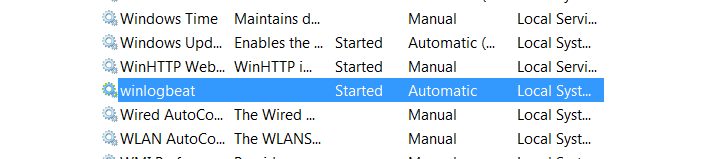
## Ниже ещё мы закомментим вывод в в elasticsearch и раскомментим вывод в наш logstash:

## 

## Осталось сконфигурировать дэшборды для винлогбита и добавить index в elasticsearch (winlogbeat-\*). Это все делается по мануалу без проблем (опять же, если все версии одинаковы, у меня была проблема с дэшбордами, т.к. Kibana 7.0 не совместима с winlogbeat 6.7.1).

## После этого можно запустить нашу созданную службу:

Services.msc



## *Либо: net start winlogbeat*

## Теперь можем посмотреть, есть ли логи в кибане:

## 

## Как видим, появился индекс winlogbeat-\* (ну и сами логи тоже на месте). Теперь разберемся с Sysmon. Встроенный логгер событий в Windows недостаточно хорош, и не подходит для нашей задачи. К счастью Microsoft сделали отличную утилиту Sysmon (скачиваем с офф сайта) которая позволяет нам, при условии правильного конфига, мониторить практически все события на машине. Хороший пример конфига файла можно скачать с

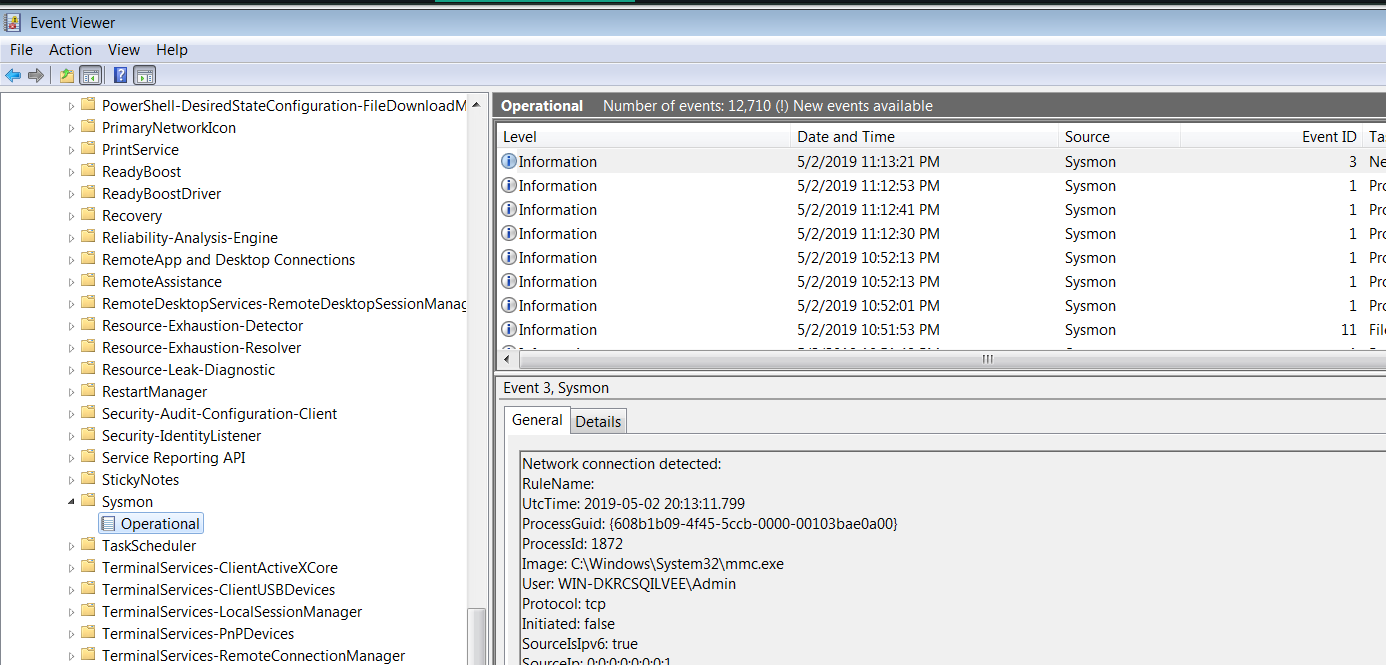
<https://raw.githubusercontent.com/SwiftOnSecurity/sysmon-config/master/sysmonconfig-export.xml>

## Конфиг очень хороший, собирается и дорабатывается регулярно. Поэтому за основу возьмем его. Процесс установки конфига в Sysmon очень прост:

## Sysmon.exe -i Sysmonconfig-export.xml -accepteula -h md5,sha256 -n -l

## 

## 



В Event Viewer появился поставщик логов Sysmon

## Да, и не забудем поставить режим авто для службы sysmon (чтобы после перезагрузки все работало).

## 

## Отлично, логи собираются Sysmon’ом. И наш конфиг для winlogbeat их уже распознает и кидает в логстеш. Осталось совсем немного, а именно доработать xml файл с учетом атаки credential dumping + настроить фильтр именно на этот ивент + настроить дэшбоард.

## 

## Для начала, (не сделал вовремя скрин, к сожалению) у меня была проблема с тем, что vbox service, процесс virtualbox’a создавал слишком много шума и преобладал в логах. С целью упрощения лучше отключить вывод ивентов связанных с этим процессом.

## Это можно поправить в конфиге sysmon-export-config.xml

## Кстати, забыл упомянуть про SYSMON и его EVENT CODES.

## Где-то выше на скрине, видно что в журнале логов у сисмона есть свои коды у ивентов. Вот к примеру Sysmon EVENT ID 10 : INTER-PROCESS ACCESS. Или скажем SYSMON EVENT ID 6 : DRIVER LOADED INTO KERNEL.

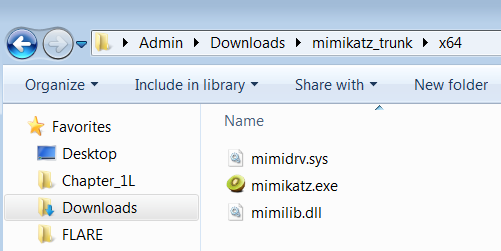
## Так вот, к чему я, нас интересует в данный момент именно ID 10. Так что добавим нужные нам строки в конфиг файл:

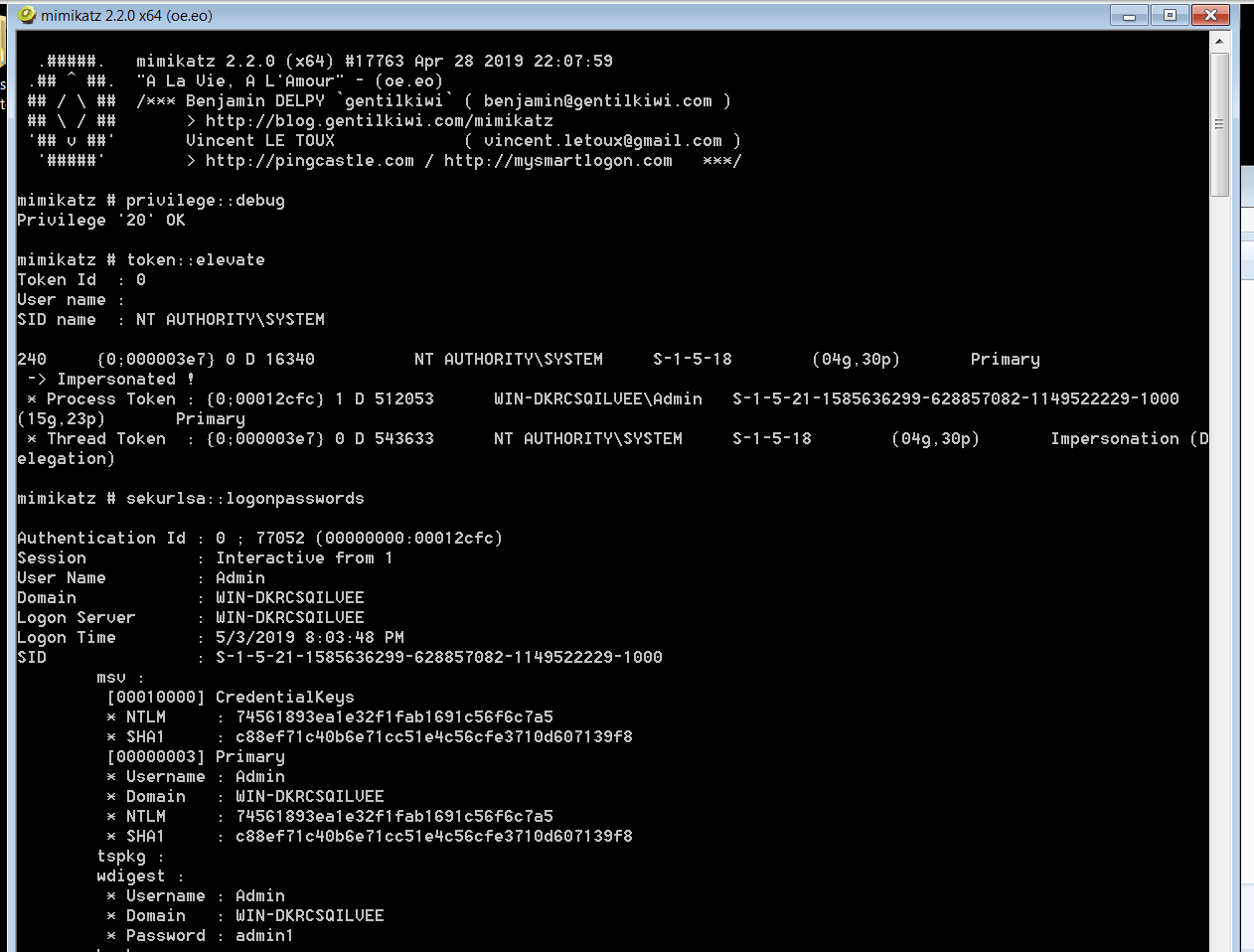
## 

## 

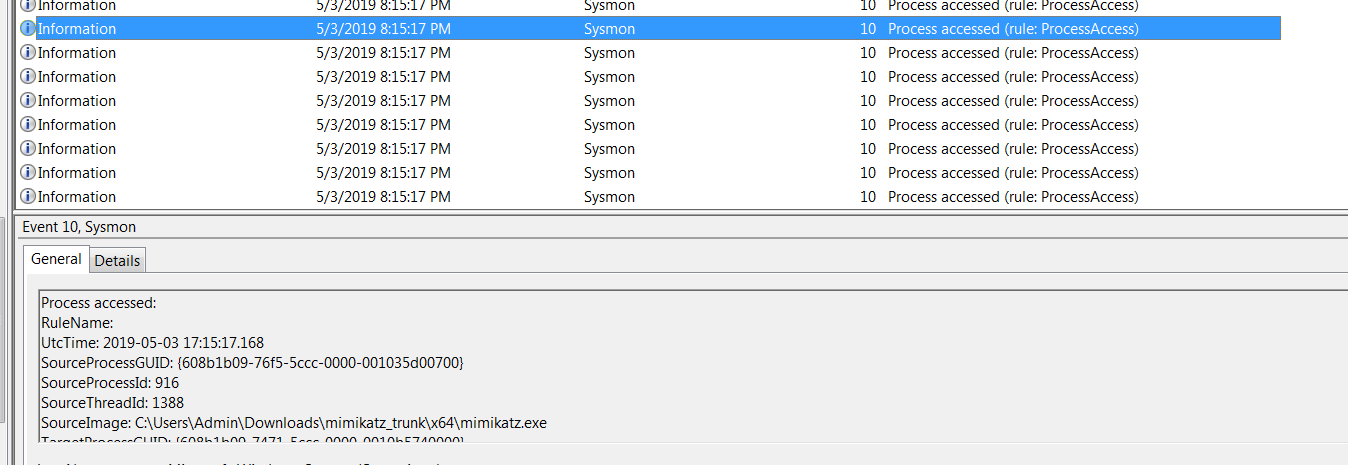
## Как видно по скрину мы избавились от ненужных мусорных уведомлений от vboxserice, а также добавили мониторинг службы lsass.exe (по заданию).

## Теперь осталось только правильно провести атаку credential dumping. Для этого нам пригодится **mimikatz** (утилита, наверное, не нуждающаяся в представлении). Для чистоты эксперимента не забудем отключить антивирусный софт и встроенный дефендер тоже выключить (можно через политики безопасности gpedit.msc). Скачиваем акутальную версию mimikatz, распаковываем и запускаем.

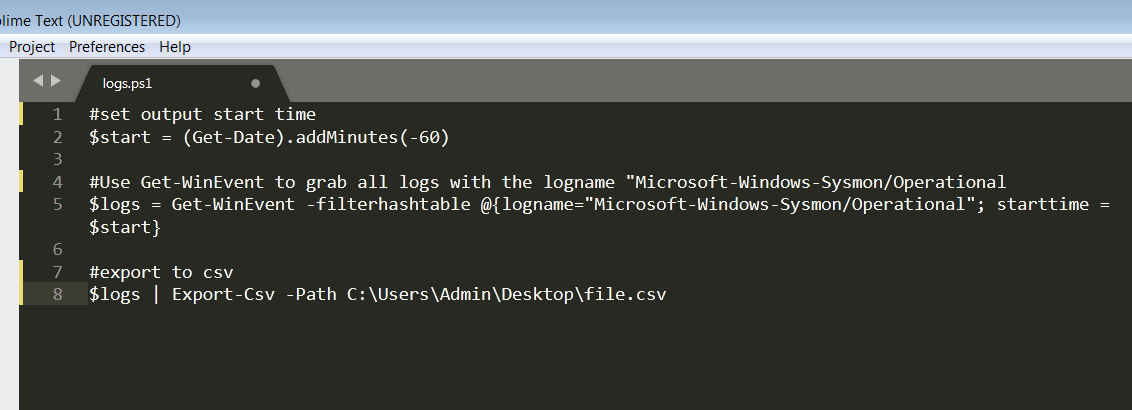
  
Mimikatz по сути извлекаем пароли пользователей winodws из памяти. Для этого запускаем утилиту с

privilege::debug   
token::elevate   
sekurlsa::logonpasswords  
  


В EventViewer’e полно событий такого вида:

  
Чтобы понять как отфильтровать и выбрать нужные нам для идентификации атаки события нужно проанализировать механизм работы mimikatz, его взаимодействие с системой. Смысл в том, что мимикатз с помощью выполненных команд (в частности SeDebugPrivilege) поднимает себе доступ чтобы читать и делать запросы процесса lssas (дамп памяти которого содержит пароли).

Написал маленький скрипт для экспорта логов из-под винды (скрипт использующий встроенный методы powershell’a экспортирующий логи в формате csv за последний час)

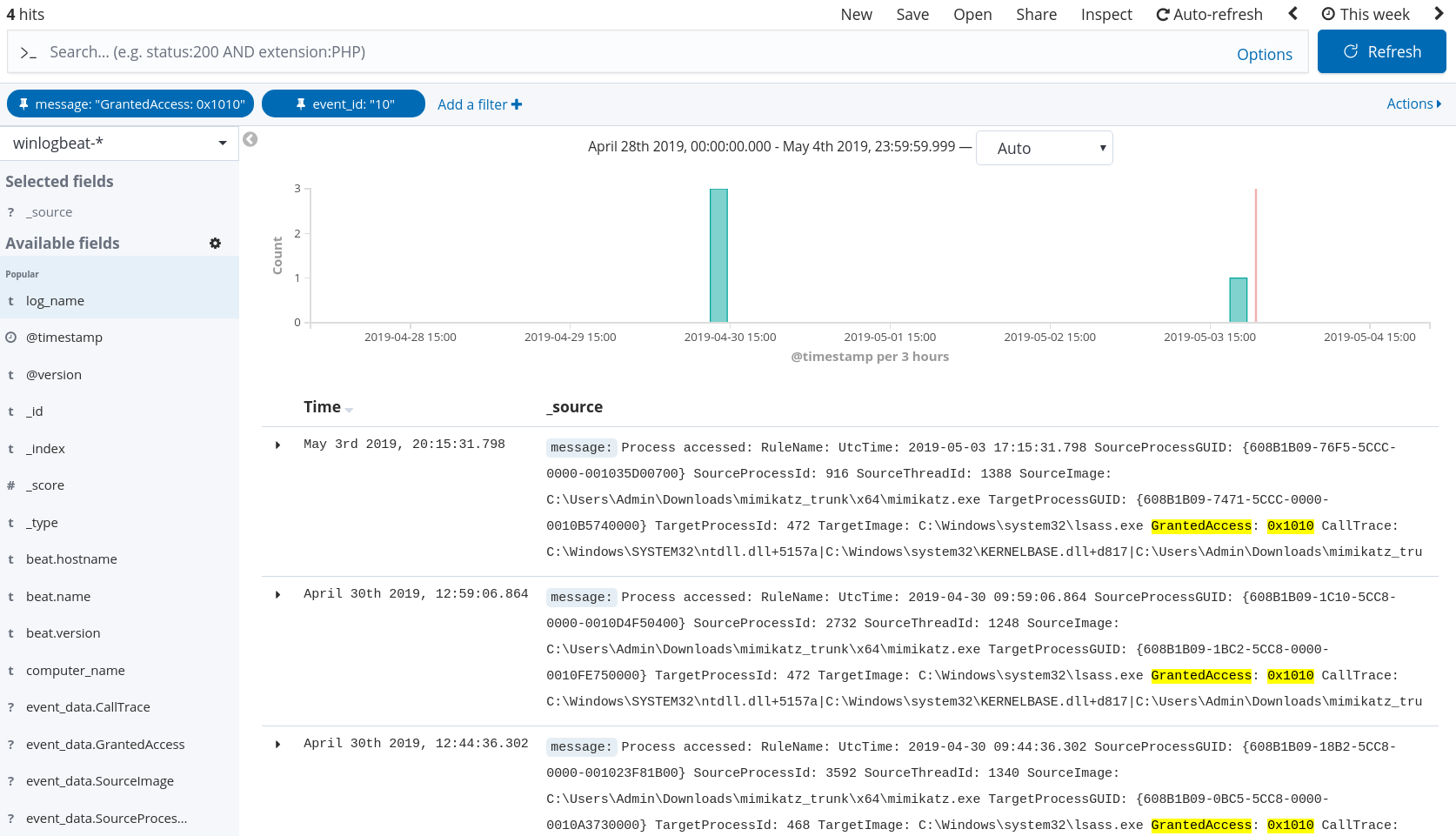
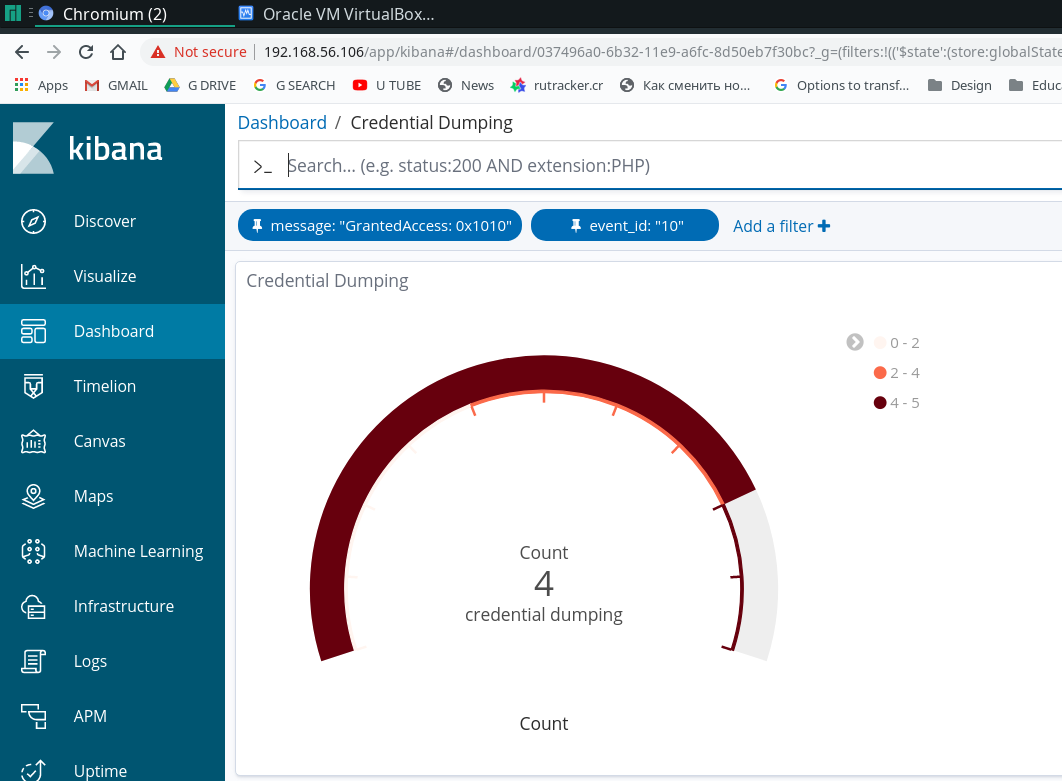


На гите есть в открытом доступе исходники функции sekurlsa\_aquireLSA где можно пронаблюдать как выглядит запрос mimikatza для lsass.exe и исходят из этого выстроить правильный фильтр для детектирования credential dumping с помощью mimikatz.

DWORD processRights = PROCESS\_VM\_READ | ((MIMIKATZ\_NT\_MAJOR\_VERSION < 6) ? PROCESS\_QUERY\_INFORMATION : PROCESS\_QUERY\_LIMITED\_INFORMATION);



PROCESS\_QUERY\_LIMITED\_INFORMATION (0x1000) – запрашивает информацию о процессе  
PROCESS\_VM\_READ (0x0010) – для чтения памяти процесса  
  
Применив битовую операцию OR получаем маску 0x1010 - выставим в кибане фильтр с GrantedAccess 0x1010 и EventID 10 (10 код потому что мы добавили в конфиг файла sysmon’а эти строки: <TargetImage condition="is">C:\Windows\system32\lsass.exe</TargetImage> );  
  
Из плюсов, мы не привязываемся к имени исполняемого файла, mimikatz.exe, а выстраиваем паттерн на метод sekurlsa. Теперь осталось проверить фильтры в кибане, и создать на их основе dashboard.

## Для такого типа атаки фильтры справляются. Но в данной лабораторной работе я рассматривал только вариант с запуском бинарника mimikatz из командной строки (типичный вариант). Думаю что надо проверить другие варианты использования атаки, к примеру используя powersploit (модуль из коллекции powershell) “Invoke-Mimikatz.ps1” (отличия этого метода в том, что мы загружаем мимикатз сразу в память используя poweshell, делая дамп кредов без записи на диск). Возможно результат будет отличаться и надо будет использовать другие фильтры в кибане. Если успею, то проведу дополнительные тесты.

## 