实训任务四：基于 Windows 事件数据分析提升系统安全

一、目的

1、了解 Windows 事件；

2、实现对 Windows 事件的数据分析。

二、学时

4 学时

扫码看实训演示

扫码看任务步骤

三、类型

综合型

四、需求

1、硬件

每人配备计算机 1 台，不低于双核 CPU、8G 内存、500GB 硬盘。

每小组配备服务器 1 台。

2、软件

Windows 操作系统，安装 puTTY 管理终端软件，安装 VMware ESXi 控制台软件。

服务器安装 VMware ESXi 7.0。

3、网络

计算机、服务器、虚拟主机使用固定 IP 地址接入局域网，并支持对互联网的访问。

4、工具

无。

五、任务要求

1、完成 Winlogbeat 的安装与配置；

2、实现 Windows 事件分析以达到 Windows 系统安全管理。

六、考核要求

1、提交《Windows 事件数据分析报告》；

2、提交 Windows 事件可视化分析成果截图/演示视频。

七、任务步骤

Windows 事件记录程序为应用程序和操作系统提供了一种标准的方式来记录重要的事

件，主要有系统（System）、安全（Security）和应用程序（Application）三种，Windows 事件可通过事件查看器查看。

任务 1：使用 Winlogbeat 采集 Windows Events 数据

Winlogbeat 使用 Windows API 读取一个或多个事件日志，并根据用户配置的条件过滤事件日志，然后将事件日志发送到指定的输出目的地（Elasticsearch 或 Logstash）。

步骤 01：获取 Winlogbeat 程序

从 https://www.elastic.co/cn/downloads/beats/winlogbeat 获取安装的 ZIP 文件。

步骤 02：安装配置 Winlogbeat

将 ZIP 文件解压到 C:\Program Files\winlogbeat-7.8.0-windows，并重命名文件夹为 Winlogbeat。

鼠标右键 Windows 开始按钮，打开 Windows PowerShell(管理员)，运行以下命令将 Winlogbeat 安装为 Windows 服务。

参考命令：

```
& 'C:\Program Files\winlogbeat\install-service-winlogbeat.ps1'
```

步骤 03：修改 Winlogbeat 配置文件

配置文件：

```
winlogbeat.event_logs:
  - name: Application
  - name: System
  - name: Security
    fields:
      indextype: "winlogbeat-172.20.1.72"
#此处省略部分信息
#output.elasticsearch:
  # Array of hosts to connect to
  #hosts: ["localhost:9200"]
#此处省略部分信息
output.logstash:
  # The Logstash hosts
  hosts: ["10.10.2.155:5044"]
#此处省略部分信息
```

步骤 04：启动 Winlogbeat

参考命令：

```
& 'C:\Program Files\winlogbeat\winlogbeat.exe ' setup
Start-Service winlogbeat
```

任务 2：使用 Logstash 进行数据清洗与格式化

步骤 01：分析 Windows 事件数据字段

创建 Logstash 配置文件将 Windows 日志输出在控制台中，配置文件路径为/etc/logstash/conf.d/logstash_winlog.conf，配置文件如下所示。

配置文件：

```
input {
  beats {
    port => 5044
```

```
  }
}
filter{
      if [indextype] == "winlog-172.20.1.72" {
            ruby {
                  path => "/etc/logstash/script/winlog.rb"
            }
      }

}
output {
    stdout { codec => rubydebug }
}
```

启动 Logstash 可在控制台中查看 Windows 事件日志字段，如表 4-1 所示。

参考命令：

```
logstash -f /etc/logstash/conf.d/logstash_winlogbeat.conf
```

表 4-1 Windows 日志主要字段

序号	字段名称	字段含义
1	event.code	事件代码
2	event.created	日志采集时间
3	host.architecture	主机架构
4	host.hostname	主机名
5	host.ip	主机 IP
6	host.mac	主机 MAC 地址
7	host.os.family	操作系统类型
8	host.os.kernel	操作系统版本
9	host.os.name	操作系统名称
10	host.os.version	操作系统版本
11	log.level	日志级别
12	message	日志描述
13	winlog.channel	日志类型
14	winlog.event_data.ProcessCreationTime	日志生成日期
15	winlog.event_data.Type	日志类型代码
16	winlog.keywords	日志关键字
17	winlog.process.pid	日志进程 ID
18	winlog.process.thread.id	日志线程 ID
19	winlog.task	任务类别

步骤 02：配置 Logstash 实现数据清洗与格式化

创建/etc/logstash/script/winlog.rb，并编辑内容。

配置文件：

```
require 'rubygems'
require 'json'
require 'time'
def register(params)

end

def filter(event)
    begin
        message=event.get('message')
        message_obj = JSON.parse(message)#转化为 json 对象
        messages = message_obj["message"]
        if message_obj["winlog"]["channel"] == "Application"
            event.set("level", message_obj["log"]["level"])
            event.set("channel", message_obj["winlog"]["channel"])
            event.set("event_id", message_obj["winlog"]["event_id"])
            event.set("provider_name", message_obj["winlog"]["provider_name"])
            event.set("computer_name", message_obj["winlog"]["computer_name"])
            event.set("record_id", message_obj["winlog"]["record_id"])
            event.set("keywords", message_obj["winlog"]["keywords"])
            event.set("task", message_obj["winlog"]["task"])
            event.set("api", message_obj["winlog"]["api"])
            event.set("Event code", message_obj["winlog"]["event_data"]["param1"])
            event.set("Event message", message_obj["winlog"]["event_data"]["param2"])
            event.set("Event time", message_obj["winlog"]["event_data"]["param3"])
            event.set("Event ID", message_obj["winlog"]["event_data"]["param5"])
            event.set("Event sequence", message_obj["winlog"]["event_data"]["param6"])
            event.set("Event occurrence", message_obj["winlog"]["event_data"]["param7"])
            event.set("Event detail code", message_obj["winlog"]["event_data"]["param8"])
        elsif message_obj["winlog"]["channel"] == "System"
            event.set("level", message_obj["log"]["level"])
            event.set("channel", message_obj["winlog"]["channel"])
            event.set("event_id", message_obj["winlog"]["event_id"])
            event.set("provider_name", message_obj["winlog"]["provider_name"])
            event.set("computer_name", message_obj["winlog"]["computer_name"])
            event.set("record_id", message_obj["winlog"]["record_id"])
            event.set("keywords", message_obj["winlog"]["keywords"])
            event.set("task", message_obj["winlog"]["task"])
            event.set("api", message_obj["winlog"]["api"])
            event.set("Event code", message_obj["winlog"]["event_data"]["param1"])
            event.set("Event message", message_obj["winlog"]["event_data"]["param2"])
        else
            event.set("level", message_obj["log"]["level"])
            event.set("channel", message_obj["winlog"]["channel"])
            event.set("event_id", message_obj["winlog"]["event_id"])
            event.set("provider_name", message_obj["winlog"]["provider_name"])
            event.set("computer_name", message_obj["winlog"]["computer_name"])
            event.set("record_id", message_obj["winlog"]["record_id"])
            event.set("keywords", message_obj["winlog"]["keywords"])
            event.set("task", message_obj["winlog"]["task"])
            event.set("api", message_obj["winlog"]["api"])
            event.set("TargetLogonId", message_obj["winlog"]["event_data"]["TargetLogonId
"])
            event.set("TargetUserName", message_obj["winlog"]["event_data"]["TargetUserN
ame"])
```

```
                event.set("LogonType", message_obj["winlog"]["event_data"]["LogonType"])
        end
        #查询时段
        local_time_e = message_obj["event"]["created"]
        local_times = local_time_e.split(/[A-Z]/)
        local_time_a = local_times[0]+" "+local_times[1]
        local_time_b = local_time_a.split(/\./)
        local_time_c = local_time_b[0]
        local_time_d = local_time_c.to_s
        local_time = Time.parse(local_time_d) + 60*60*8
        event.set("local_time", local_time)
        event.set('[local_time_arr][year]', local_time.year)
        if(local_time.month.to_s.length==1)
                event.set('[local_time_arr][month]', "0"+local_time.month.to_s)
        else
                event.set('[local_time_arr][month]', local_time.month.to_s)
        end
        if(local_time.day.to_s.length==1)
                event.set('[local_time_arr][day]', "0"+local_time.day.to_s)
        else
                event.set('[local_time_arr][day]', local_time.day.to_s)
        end
        event.set('[local_time_arr][wday]', local_time.wday)
        event.set('[local_time_arr][yday]', local_time.yday)

        if(local_time.hour.to_s.length==1)
                event.set('[local_time_arr][hour]', "0"+local_time.hour.to_s)
        else
                event.set('[local_time_arr][hour]', local_time.hour.to_s)
        end
        event.set('[local_time_arr][min]', local_time.min)
        event.set('[local_time_arr][sec]', local_time.sec)
    end
    return [event]
end
```

步骤 03：推送数据到 Elasticsearch

创建 Logstash 配置文件将 Windows 日志输出在 Elasticsearch 中，配置文件如下所示。

配置文件：

```
input {
  beats {
    port => 5044
  }
}
filter{
    if [indextype] == "winlog-172.20.1.72" {
        ruby {
            path => "/etc/logstash/script/winlog.rb"
        }
    }

}
output {
  if [indextype]== "winlog-172.20.1.72" {
    elasticsearch{
      hosts => ["10.10.2.152:9200"]
```

```
        index => "winlog-172.20.1.72-%{+YYYY.MM.dd}"
    }
  }
}
```

启动 Logstash。

参考命令：

```
logstash -f /etc/logstash/conf.d/logstash_winlogbeat.conf &
```

任务 3：在 Kibana 上创建分析模型并进行分析

步骤 01：设计分析模型

根据数据格式创建分析模型，如表 4-2 所示。

表 4-2 Windows Event 分析模型

序号	分析字段	分析模型	图表类型
1	Event ID	Window 事件总数	指标
2	Event ID+ level	Windows 错误事件总数	指标
3	Event ID+local_time	Windows 事件日志变化趋势	折线图
4	Level	Windows 事件日志类别占比	饼图
5	Event code	Windows 事件日志 ID 排行	柱状图
6	local_time_arr.hour	Windows 一天中每小时事件日志量	柱状图
7	local_time_arr.day	Windows 一月中每天事件日志量	柱状图
8	keywords	Windows 事件关键字占比	柱状图

步骤 02：创建分析模型

依据表 4-2 实现图表可视化，如图 4-1 所示。

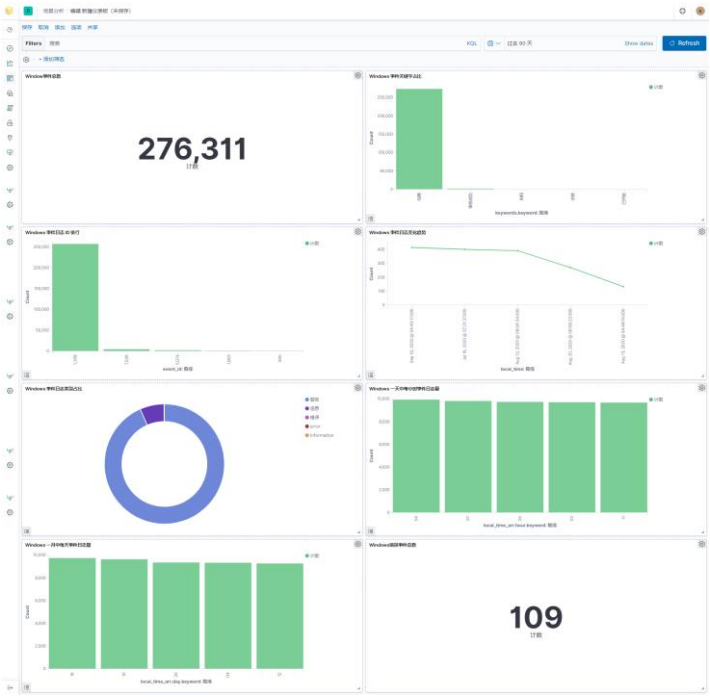

图 4-1 分析模型可视化

八、实验思考

1、深入认识 Beats。

（1）Beats 有几种，分别用于哪些场景?

（2）Winlogbeat 支持的数据有哪些？

（3）Elastic Stack 与 ELK 的关系是什么？

2、Logstash 的用途和容灾。

（1）Logstash 的作用是什么？

（2）Logstash 支持的数据操作有哪些？

（3）Logstash 没有安全的终止时，管道中未处理的数据会怎么处理？会造成数据丢失么？

3、Windows 事件数据分析。

（1）Windows 事件有哪些类型？分别都代表什么？

（2）如果获取更多的 Windows 事件数据？

（3）Windows 事件分析结果能够发现哪些系统风险？