使用Security Onion构建安全监控平台

al0ne

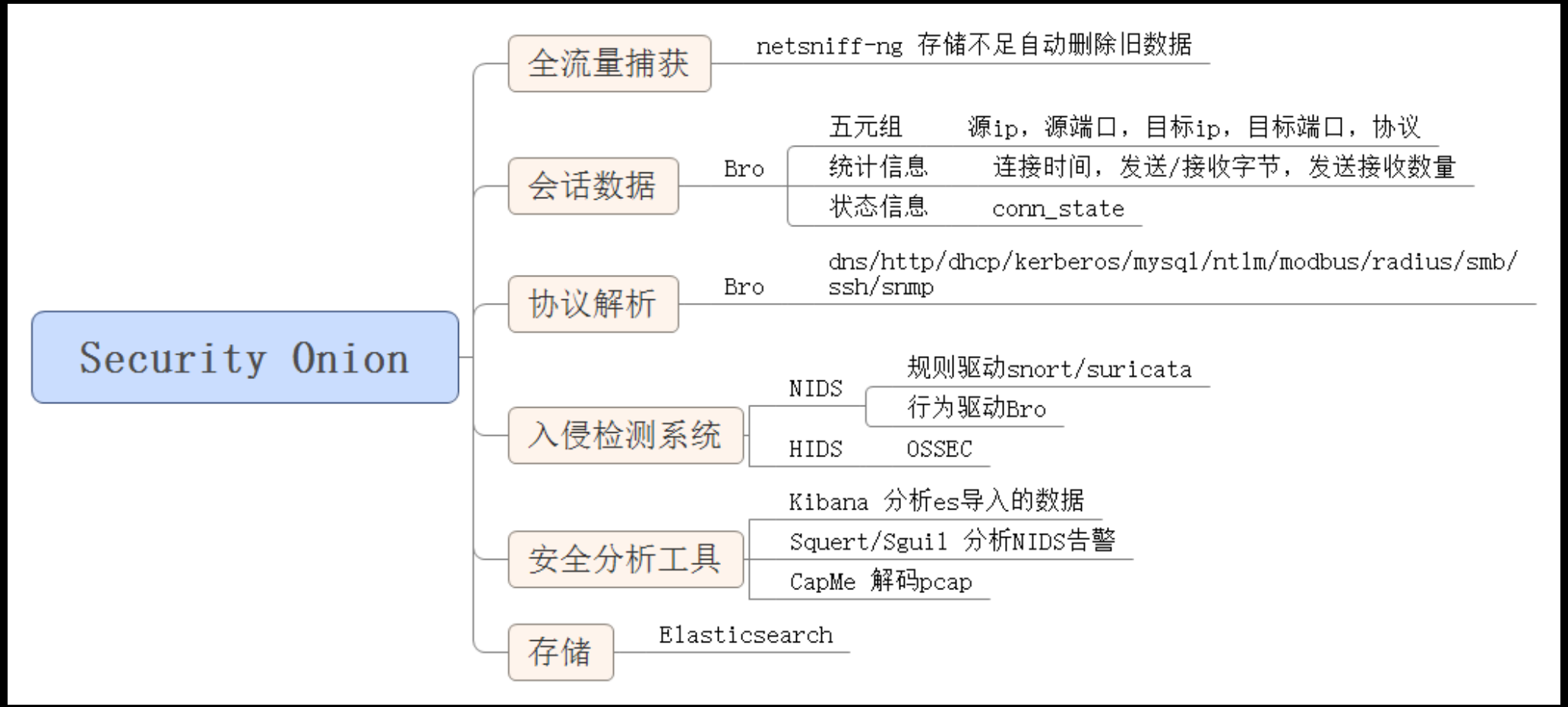
概述

- Security Onion（安全洋葱）是一个开源免费用来做企业入侵检测\网络安全监控/日志管理的Linux发行版，基于Ubuntu 16.04系统，集成了常见的入侵检测工具suricata/snort/ossec/bro、ELK、安全分析Sguil/Squert等工具

部署模式

- 单传感器（收集，存储，检测，分析）
- 服务器（存储，分析）+传感器（收集，检测）

数据来源


Security Onion
全流量捕获
netsniff-ng 存储不足自动删除旧数据
会话数据
Bro
五元组 源ip，源端口，目标ip，目标端口，协议
统计信息 连接时间，发送/接收字节，发送接收数量
状态信息 conn_state
协议解析
Bro
dns/http/dhcp/kerberos/mysql/ntlm/modbus/radius/smb/ssh/snmp
入侵检测系统
NIDS
规则驱动snort/suricata
行为驱动Bro
HIDS
OSSEC
安全分析工具
Kibana 分析es导入的数据
Squert/Sguil 分析NIDS告警
CapMe 解码pcap
存储
Elasticsearch

netsniff-ng 全流量捕获

- linux下高性能全流量抓包工具， 以pcap包格式存储流量数据
  - /nsm/sensor_data/eth0/dailylogs/YYYY-MM-DD/
- 空间接近90%自动删除旧数据
  - /nsm/sensor_data/xxx/dailylogs 全流量数据
  - /nsm/bro/logs bro 日志目录
  - /nsm/sensor_data/xxx/snort snort告警日志
- TrimPCAP
  精简抓取的全包流量，将pcap文件每个流只保留前*个字节
  sudo find /nsm/sensor_data/*/dailylogs/*/* -mtime +7 -type f -exec /opt/trimpcap.py 102400 {} \;
  7天前的pcap文件每个流只保留前100kb

Suricata IDS

- Suricata是一个开源高性能基于规则的入侵检测系统
- 特点：
  多线程
  支持pf-ring抓包模式(linux内核高速数据包捕获库)
  支持Intel发布的高性能正则表达式匹配库Hyperscan
  完全兼容Snort规则、Lua脚本、YARA（官方还在开发，将流量中的文件提取出来，然后调用yara规则去匹配）
  离线分析pcap包
  能够解析HTTP、SMTP、DNS、TLS等协议
- 官网：https://suricata-ids.org/

Suricata IDS

- Suricata.yaml配置文件：
  - HOME_NET: “[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]” 配置内网地址
  - EXTERNAL_NET: "any”
  - default-rule-path: /var/lib/suricata/rules
  - rule-files:
- xxx.rules
  - alert:
payload-buffer-size: 6kb
http-body-printable: yes
http/tls/dns/dns/files/smb/flow
  - checksum-checks: no　　关闭校验和

Suricata IDS 命令行

- suricata
  -c /etc/suricata/suricata.yaml 选择配置文件
  -i eth0 选择监听接口
  -r 读取本地pcap包
  -l 将结果保存到设置的目录，默认是配置文件里/var/log/suricata目录
  --build-info 查看编译信息，检查有那些库没有启用
  - T 配置文件 测试模式，加载配置文件测试，检查配置是否存在错误
  - F 过滤文件 在抓包时通过BPF包过滤来排除一些不需要的流量
  - D 守护进程方式启动
  --pfring-int=ens3 --pfring-cluster-id=99 --pfring-cluster-type pfring方式抓包
  --list-runmodes 查看支持的运行模式（例如查看pfring是否已经启用）

Suricata IDS 规则集

- 规则来源：
  EmergingThreats&ET pro规则（收费）官方自带规则，覆盖较广，主要检测一些扫描行为/病毒木马/蠕虫/协议特征/攻击特征/恶意ip/shellcode等
  ptresearch/attackdetection 包含了近几年cve漏洞的检测规则
  sslbl/ssl-fp-blacklist 恶意软件与僵尸网络的ssl证书特征
  suricata lua rule github上使用lua语言写的规则
- 规则更新管理：
  suricata-update ： https://github.com/OISF/suricata-update
  scirius： https://github.com/StamusNetworks/scirius

Suricata IDS 规则编写

WannaCry匹配规则示例

#规则头 alert smb $HOME_NET any -> any any (msg:“ET EXPLOIT Possible ETERNALBLUE MS17-010 Echo Response”; flow:from_server,established; content:“|00 00 00 31 ff|SMB|2b 00 00 00 00 98 07 c0|”; depth:16; content:“|4a 6c 4a 6d 49 68 43 6c 42 73 72 00|”; distance:0; flowbits:isset,ET.ETERNALBLUE; metadata: former_category EXPLOIT; classtype:trojan-activity; sid:2024218; rev:2; metadata:attack_target SMB_Server, deployment Internal, signature_severity Critical, created_at 2017_04_17, updated_at 2017_11_27;) #规则体

Suricata IDS 规则编写

- 规则头 alert smb $HOME_NET any -> any any
  - 告警动作 pass跳过匹配 drop ips模式拒绝 reject 主动拒绝 alert 记录
  - 协议 TCP UDP ICMP IP（TCP+UDP） http ftp smb dns
  - 源ip/源端口 1.1.1.1 ,1.1.1.1/24,$HOME_NET,[1.1.1.1,2.2.2.2]
  - ->单向/<>双向流量
  - 目标ip/目标端口

Suricata IDS 规则编写

- 规则体
  - msg ： 规则名称
  - content：匹配内容（字符串，十六进制，pcre正则，偏移控制，协议修饰符）
  - sid：唯一id,不可重复
  - rev：版本，默认1然后依次递增
  - reference：引用来源
  - classtype：规则类型
  - threshold：阀值
  - flow字段

Suricata IDS 规则编写

- content 内容匹配
  - content:”匹配内容”;控制条件1；控制条件2；
  - content:”root”;nocase；　　#匹配字符串内容，nocase不区大小写
  - content:”|FF|SMB|2B|”　　#16进制与字符串结合
  - content:“xss”;offset 100;　　#从开始位置偏移100个字节后匹配
  - content:“xss”;offset 100;depth 200;　#匹配第100个字节到200字节之间的数据
  - content:“msg1”;content:“msg2”;distance 25;　#第一个结果位置偏移25字节后匹配
  - content:”msg1”; content:”msg2”; distance:1;within:7; #第一个结果位置偏移1~7字节内
  - content:”xss”; fast_pattern; 快速匹配

Suricata IDS 规则编写

- flow流控制
  - 特定时间内五元组相同属于同一个流（1.根据fin或者rst，2.空闲超时）
  - to_client/from_server 服务器到客户端流量
  - to_server/from_client 客户端到服务器流量
  - established 匹配已建立连接流量（tcp通过三次握手，udp通过双向流量）
  - no_stream 不匹配流重组的包
  - flowbits set , name 规则a设置条件，
  - flowbits isset, name 规则b 选择条件，一条流记录中同时命中2条规则才进行告警

Suricata IDS 规则编写

- threshold阀值: type <threshold|limit|both>, track <by_src|by_dst>, count <N>, seconds <T>
  - type threshold 最小阀值匹配到多少次数才进行告警，type limit 限制告警次数
  - track 来源
  - count 告警次数
  - seconds 给出一个时间范围
  - threshold: type threshold, track by_src, count 5, seconds 120;

Suricata IDS 检测案例

- 案例1：通过阀值type threshold 检测mysql暴力破解
- alert tcp $HOME_NET 3306 -> $EXTERNAL_NET any (msg:"ET SCAN Multiple MySQL Login Failures Possible Brute Force Attempt"; flow:from_server,established; dsize:<251; byte_test:1,<,0xfb,0,little; content:"|ff 15 04 23 32 38 30 30 30|"; offset:4; threshold: type threshold, track by_src, count 5, seconds 120; metadata: former_category SCAN; reference:url,doc.emergingthreats.net/2010494; classtype:attempted-recon; sid:2010494; rev:4; metadata:created_at 2010_07_30, updated_at 2017_05_11;)

MySQL ERROR 1045 (28000): Access denied for user

Suricata IDS 告警

- 告警信息调整
  - 禁用规则 /etc/nsm/pulledpork/disablesid.conf
    2101411　根据规则id禁用规则
    pcre:ET MISC 用正则匹配
  - 排除指定ip /etc/nsm/rules/threshold.conf
    suppress gen_id 1, sig_id 2101411, track by_src, ip 172.16.1.1/24

Suricata IDS 规则编写

- 流量中提取文件
  - 在suricata.yaml文件里面开启文件存储功能

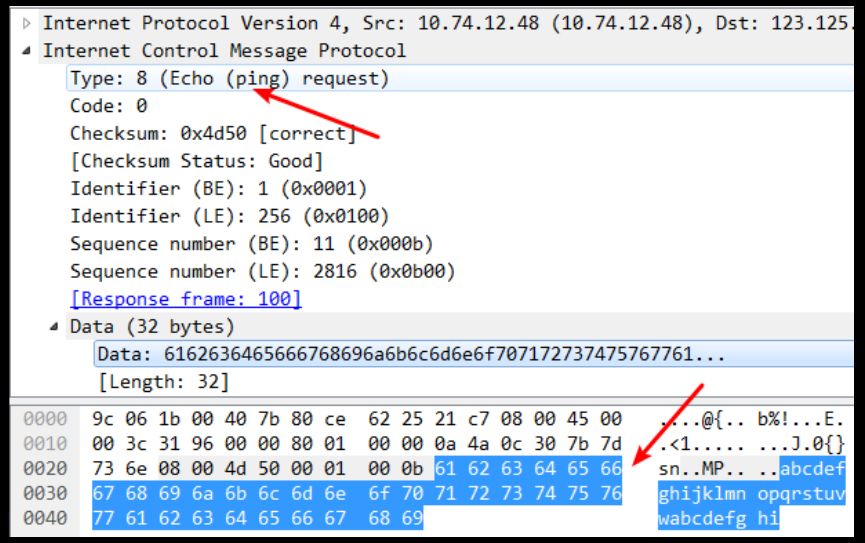
```
- file-store:
    enabled: yes       # se
    log-dir: files     # di
    force-magic: no    # fo
    # force logging of che
    # sha1 and sha256
    #force-hash: [md5]
    force-filestore: no #
    # override global stre
    # perform file extract
    #stream-depth: 0
    waldo: file.waldo # wa
    # uncomment to disable
    write-meta: yes
```

对应的规则

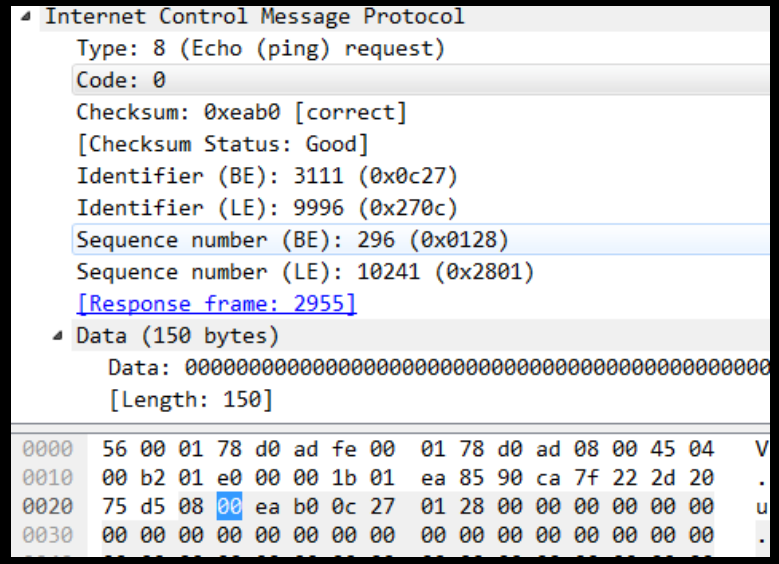
alert http any any -> any any (msg:"File store all"; flow:established; fileext:”pdf"; filestore; sid:1; rev:1;)

Suricata IDS 检测案例

- 案例2：检测ICMP通道
  正常icmp请求，windows下date填充是字母，linux下是数字



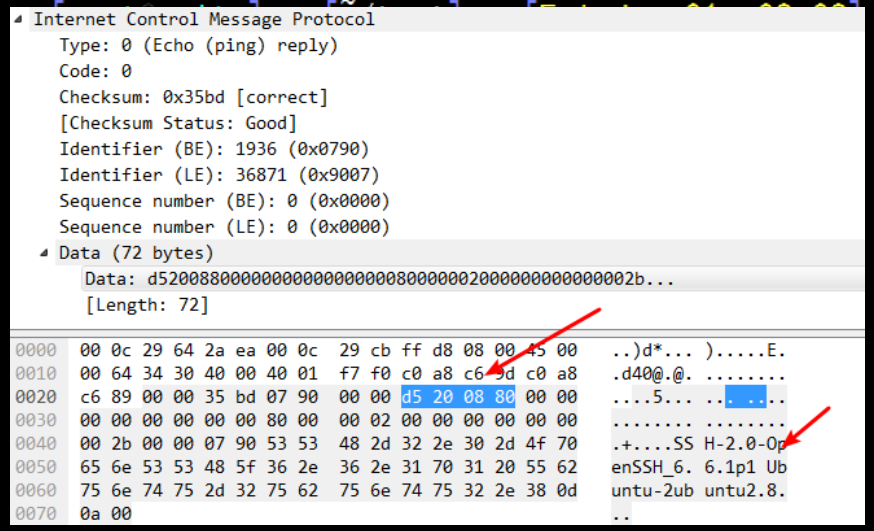
nmap扫描发出的icmp请求



Suricata IDS 检测案例

- 案例2：检测ICMP通道
  suricata –c /etc/suricata/suricata.yaml –r ptunnel.pcap –l ./

```
└─[$] <> find * -name 'eve.json*' |xargs -i{} cat {} | jq -c '[.src_ip,.dest_ip,
.proto,.alert.signature]' |grep -v null | sed 's/"//g;s/\[//;s/\]//' | awk -F=",
" '{a[$1$3]++;}END{for (i in a)print i, " | Alertas: "a[i];}'
192.168.198.137,192.168.198.157,ICMP,ET INFO PTUNNEL OUTBOUND  | Alertas: 117
192.168.198.157,192.168.198.137,ICMP,ET INFO PTUNNEL INBOUND  | Alertas: 3715
192.168.198.137,192.168.198.157,ICMP,ET TROJAN OpenSSH in ICMP Payload - Possibl
e Covert Channel  | Alertas: 3
```



BRO 网络流量分析

- BRO是一个开源基于异常行为检测的安全监控软件
- 特点：
  - 异常检测
  - 协议解析
  - 联动沙箱，将提取的文件通过调用cuckoo沙箱进行检测

BRO 网络流量分析

- BRO命令行
- bro
  - -r 读取一个pcap进行分析
  - -i ens3 选择监听接口
  - -C 禁用校验和
  - -f 捕获流量时进行过滤

BRO conn.log日志

bro -i ens3 local -C 开启bro监听

诊断日志

capture_loss.log

loaded_scripts.log

stats.log

packet_filter.log

会话日志

conn.log

告警信息

weird.log 协议错误

notice.log bro脚本产生的告警

协议解析日志

dns.log

files.log

http.log

sip.log

snmp.log

ssh.log

ssl.log

x509.log

BRO conn.log日志

conn.log 会话日志

第一个数据包时间 ts　uid 每个连接都会有一个uid，提供关联分析
源ip id.orig_h　　源端口id.orig_p　　目标ip：id.resp_h　目标端口id.resp_p

协议proto　服务service

连接时间 duration 发送字节数 orig_bytes　接受字节 resp_bytes

会话连接状态 conn_state

```
#open   2018-06-05-02-40-03
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       proto   service duration        orig_bytes
      resp_bytes      conn_state      local_orig      local_resp      missed_bytes    history orig_pkts       orig_ip_bytes   resp_pk
ts       resp_ip_bytes   tunnel_parents  orig_l2_addr    resp_l2_addr
#types  time    string  addr    port    addr    port    enum    string  interval        count   count   string  bool    bool    count
   string  count   count   count   count   set[string]     string  string
1528166391.974263       CqygEM2zS5cOB7uRI7      144.202.127.34  22022   123.59.170.204  53220   tcp     -       6.094414        300
     0       SF      -       -       0       DafF    6       540     4       160     -       56:00:01:6c:16:26       fe:00:01:6c:16:2
6
1528166398.193190       C1KqNh1OXUKLzW2S97      123.59.170.204  53221   144.202.127.34  22022   tcp     -       0.005752        0
       0       SF      -       -       0       FfA     2       80      1       40      -       fe:00:01:6c:16:26       56:00:01:6c:16
:26
1528166423.209139       CI17b42dIj6lu3ymec      146.0.77.159    38353   144.202.127.34  22      tcp     -       0.000086        0
       0       REJ     -       -       0       Sr      1       60      1       40      -       fe:00:01:6c:16:26       56:00:01:6c:16
:26
1528166408.551880       CaQt692qtOwpLU5zDb      144.202.127.34  40918   108.61.10.10    53      udp     dns     2.191780        135
     173     SF      -       -       0       Dd      3       219     3       257     -       56:00:01:6c:16:26       fe:00:01:6c:16:2
6
```

BRO 会话连接状态

- S0　看到连接尝试，没有回复。如果开启了防火墙，数据包被丢弃时就是S0状态
- S1　连接已建立，未终止。
- SF　正常建立和终止，三次握手与四次挥手以完成。流统计信息完整
- REJ　连接尝试被拒绝，对应着TCP扫描，端口关闭时的场景，服务器响应RST请求。
- S2　已建立连接，并看到发起者关闭尝试（但未回复响应者）。
- S3　建立连接并且看到响应者关闭尝试（但没有发起者的回复）。
- RSTO　建立连接，发起者中止（发送RST）。
- RSTR　连接成立，响应者发送了RST。
- RSTOS0　发起者发送了一个SYN后跟一个RST，我们从未看到响应者的SYN-ACK。
- RSTRH　响应者发送了一个SYN ACK，然后是一个RST，我们从未看到来自（声称的）始发者的SYN。
- SH　发起者发送SYN后跟一个FIN，我们从未看到响应者的SYN ACK（因此连接“打开了一半”）。
- SHR　响应者发送了一个SYN ACK后跟一个FIN，我们从来没有从发起者看到SYN。
- OTH　没有SYN出现，仅仅是中游流量（一个“部分连接”，后来没有关闭）。

BRO conn_state

- S0状态尝试连接没有响应
  1.请求包被防火墙丢弃，没有任何响应

sip	sport	dip	dport	Protocol	Length	Server Name	Host	Info
[redacted]	28907	[redacted]	4444	TCP	74			28907 → 4444 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSval=3275873408 TSecr=0
[redacted]	28907	[redacted]	4444	TCP	74			[TCP Retransmission] 28907 → 4444 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSval=3275876
[redacted]	28907	[redacted]	4444	TCP	74			[TCP Retransmission] 28907 → 4444 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSval=3275879
[redacted]	28907	[redacted]	4444	TCP	74			[TCP Retransmission] 28907 → 4444 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSval=3275882
[redacted]	28907	[redacted]	4444	TCP	74			[TCP Retransmission] 28907 → 4444 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSval=3275886
[redacted]	28907	[redacted]	4444	TCP	74			[TCP Retransmission] 28907 → 4444 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSval=3275889
[redacted]	33566	[redacted]	2222	TCP	74			33566 → 2222 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSval=3275898849 TSecr=0
[redacted]	33566	[redacted]	2222	TCP	74			[TCP Retransmission] 33566 → 2222 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSval=3275902

2.进行UDP端口扫描时，端口未开放时，会响应type 3 code 3 ICMP端口不可达

No.	Time	sip	sport	dip	dport	Protocol	Length
2947	15650.182732	185.200.118.72	34796	144.202.127.34	1194	OpenVPN	56
2948	15650.182856	144.202.127.34	34796	185.200.118.72	1194	ICMP	84

```
▷ Frame 2948: 84 bytes on wire (672 bits), 84 bytes captured (672 bits)
▷ Ethernet II, Src: 56:00:01:6c:16:26 (56:00:01:6c:16:26), Dst: fe:00:01:6c:16:26 (fe:00:01:6c:16:26)
▷ Internet Protocol Version 4, Src: 144.202.127.34 (144.202.127.34), Dst: 185.200.118.72 (185.200.118.72)
◢ Internet Control Message Protocol, Opcode: P_CONTROL_HARD_RESET_CLIENT_V2, Key ID: 0
    Type: 3 (Destination unreachable)
    Code: 3 (Port unreachable)
    Checksum: 0xf007 [correct]
    [Checksum Status: Good]
    Unused: 00000000
  ▷ Internet Protocol Version 4, Src: 185.200.118.72 (185.200.118.72), Dst: 144.202.127.34 (144.202.127.34)
  ▷ User Datagram Protocol, Src Port: 34796, Dst Port: 1194
  ▷ OpenVPN Protocol
```

BRO conn_state

- REJ 尝试连接被拒绝，服务器响应RST请求，常见TCP端口扫描，端口未开放

sip	sport	dip	dport	Protocol	Length	Server Name	Host	Info
71.6.146.185	31119	[redacted]	4444	TCP	58			31119 → 4444 [SYN] Seq=0 Win=46573 …
[redacted]	4444	71.6.146.185	31119	TCP	54			4444 → 31119 [RST, ACK] Seq=1 Ack=1…

- bro-cut -d ts id.orig_h id.orig_p id.resp_h id.resp_p service duration orig_bytes resp_bytes conn_state<conn.log|awk '$10=="REJ"'

2018-06-03T17:12:45+0000	189.152.173.174	19681	[redacted]	23	-	0.000209	0	0	REJ
2018-06-03T17:13:37+0000	188.16.17.106	31293	[redacted]	23	-	0.000070	0	0	REJ
2018-06-03T17:13:47+0000	185.222.209.108	39828	[redacted]	22	-	0.000132	0	0	REJ
2018-06-03T17:14:06+0000	71.6.146.185	31119	[redacted]	4444	-	0.000184	0	0	REJ
2018-06-03T17:14:14+0000	201.148.96.21	60079	[redacted]	445	-	0.000065	0	0	REJ
2018-06-03T17:14:14+0000	201.148.96.21	60079	[redacted]	445	-	0.000051	0	0	REJ
2018-06-03T17:14:34+0000	181.214.87.34	40915	[redacted]	7994	-	0.199151	0	0	REJ
2018-06-03T17:14:50+0000	194.67.37.90	54658	[redacted]	445	-	0.000069	0	0	REJ
2018-06-03T17:14:51+0000	194.67.37.90	54658	[redacted]	445	-	0.000086	0	0	REJ
2018-06-03T17:14:54+0000	92.63.193.75	42325	[redacted]	5113	-	0.153219	0	0	REJ
2018-06-03T17:17:07+0000	209.119.236.21	61262	[redacted]	445	-	0.000158	0	0	REJ
2018-06-03T17:17:08+0000	209.119.236.21	61262	[redacted]	445	-	0.000033	0	0	REJ
2018-06-03T17:17:22+0000	45.227.253.242	46566	[redacted]	14688	-	0.142356	0	0	REJ
2018-06-03T17:17:26+0000	184.105.139.87	38532	[redacted]	23	-	0.000066	0	0	REJ
2018-06-03T17:18:18+0000	185.222.210.61	41250	[redacted]	3551	-	0.598266	0	0	REJ
2018-06-03T17:19:09+0000	181.214.87.30	59416	[redacted]	9687	-	0.345191	0	0	REJ
2018-06-03T17:19:16+0000	77.72.82.22	58188	[redacted]	10760	-	0.178457	0	0	REJ
2018-06-03T17:19:47+0000	185.208.208.198	52612	[redacted]	3382	-	0.269841	0	0	REJ
2018-06-03T17:19:52+0000	185.222.211.102	45925	[redacted]	5222	-	0.228329	0	0	REJ
2018-06-03T17:20:07+0000	116.105.186.144	62703	[redacted]	445	-	0.000060	0	0	REJ
2018-06-03T17:20:08+0000	116.105.186.144	62703	[redacted]	445	-	0.000141	0	0	REJ
2018-06-03T17:20:59+0000	185.232.28.195	46497	[redacted]	9660	-	0.191189	0	0	REJ
2018-06-03T17:21:40+0000	141.0.176.21	41052	[redacted]	9090	-	0.217044	0	0	REJ
2018-06-03T17:22:57+0000	5.140.132.149	4798	[redacted]	23	-	0.000250	0	0	REJ
2018-06-03T17:24:12+0000	104.236.163.115	35240	[redacted]	8001	-	0.000212	0	0	REJ
2018-06-03T17:24:18+0000	92.63.193.75	42325	[redacted]	5138	-	0.151157	0	0	REJ
2018-06-03T17:24:48+0000	185.222.209.108	51919	[redacted]	22	-	0.000099	0	0	REJ
2018-06-03T17:25:05+0000	181.214.87.247	45643	[redacted]	3589	-	0.178525	0	0	REJ
2018-06-03T17:25:13+0000	180.97.106.39	33439	[redacted]	21	-	0.000094	0	0	REJ

BRO conn_state

- 发起445扫描排行前10的ip
  - bro-cut -d ts id.orig_h id.orig_p id.resp_h id.resp_p service duration orig_bytes resp_bytes conn_state<conn.log|awk '$10=="REJ"'|awk '$5 ==445'|awk '{print $2}'|sort|uniq -c |sort -nr|head -n 10

```
└─[$] <> bro-cut -d ts id.orig_h id.orig_p id.resp_h id.resp_p service duration orig_bytes resp_bytes conn_state<conn.log|awk '$10=="
REJ"'|awk '$5 ==445'|awk '{print $2}'|sort|uniq -c |sort -nr|head -n 10
     27 144.202.94.143
      8 122.10.99.30
      6 85.105.7.172
      6 77.87.77.140
      6 207.35.211.2
      6 187.236.145.91
      6 186.72.88.254
      6 1.196.4.8
      6 115.238.29.216
      6 110.36.229.179
```

BRO 异常检测

- 案例4：通过bro脚本检测dns隧道

  dnscat2 https://github.com/iagox86/dnscat2

192.168.198.137	43043	192.168.198.2	53	DNS	168	Standard query 0xd573 CNAME 91cc0380407bb03cd5a4010000f2df4944a7884a1070f8be3a40e85249c2.1649181aec753e3f1ce7860fa10…
192.168.198.2	53	192.168.198.137	43043	DNS	282	Standard query response 0xd573 CNAME 91cc0380407bb03cd5a4010000f2df4944a7884a1070f8be3a40e85249c2.1649181aec753e3f1c…
192.168.198.137	28843	192.168.198.2	53	DNS	138	Standard query 0x9346 TXT 9083008040ab1a45582b9d0001963400ba234ad3806d5af297413b735ab1.1af94689.1.eej.me
192.168.198.2	53	192.168.198.137	28843	DNS	185	Standard query response 0x9346 TXT 9083008040ab1a45582b9d0001963400ba234ad3806d5af297413b735ab1.1af94689.1.eej.me TXT
192.168.198.137	10559	192.168.198.2	53	DNS	103	Standard query 0x53f1 MX df4601804053709f25c5830002ee410664.1.eej.me
192.168.198.2	53	192.168.198.137	10559	DNS	154	Standard query response 0x53f1 MX df4601804053709f25c5830002ee410664.1.eej.me MX 10 b8370180405ba4ec6aed45ffffff51b2…
192.168.198.137	52272	192.168.198.2	53	DNS	103	Standard query 0x08f5 MX 584e018040b7db680a427100038c6aec74.1.eej.me
192.168.198.2	53	192.168.198.137	52272	DNS	154	Standard query response 0x08f5 MX 584e018040b7db680a427100038c6aec74.1.eej.me MX 10 ef02018040b434962924b4ffffff51b2…
192.168.198.137	3274	192.168.198.2	53	DNS	103	Standard query 0xfaa0 TXT 14aa018040eba83692b303000495bc41c8.1.eej.me
192.168.198.2	53	192.168.198.137	3274	DNS	150	Standard query response 0xfaa0 TXT 14aa018040eba83692b303000495bc41c8.1.eej.me TXT
192.168.198.137	44576	192.168.198.2	53	DNS	103	Standard query 0xe5ba MX b0f6018040d645f922993800058d640dce.1.eej.me
192.168.198.2	53	192.168.198.137	44576	DNS	154	Standard query response 0xe5ba MX b0f6018040d645f922993800058d640dce.1.eej.me MX 10 5854018040c7d7f64aed7bffffff51b2…
192.168.198.137	12821	192.168.198.2	53	DNS	103	Standard query 0xdfbb TXT 5dd1018040f15153c8046f0006f61d9cef.1.eej.me
192.168.198.2	53	192.168.198.137	12821	DNS	150	Standard query response 0xdfbb TXT 5dd1018040f15153c8046f0006f61d9cef.1.eej.me TXT
192.168.198.137	19927	192.168.198.2	53	DNS	103	Standard query 0xbf89 MX 2135018040f3194790105e0007172b3125.1.eej.me
192.168.198.2	53	192.168.198.137	19927	DNS	154	Standard query response 0xbf89 MX 2135018040f3194790105e0007172b3125.1.eej.me MX 10 272a0180409778af9a9f9dffffff51b2…
192.168.198.137	56269	192.168.198.2	53	DNS	103	Standard query 0x0de0 TXT 38d7018040f6361c0d15870008fbf2b18f.1.eej.me
192.168.198.2	53	192.168.198.137	56269	DNS	150	Standard query response 0x0de0 TXT 38d7018040f6361c0d15870008fbf2b18f.1.eej.me TXT
192.168.198.137	17743	192.168.198.2	53	DNS	103	Standard query 0x1962 CNAME f830018040912dca08b6d600092182ab37.1.eej.me
192.168.198.2	53	192.168.198.137	17743	DNS	152	Standard query response 0x1962 CNAME f830018040912dca08b6d600092182ab37.1.eej.me CNAME a7ea018040d98e4a53de07ffffff5…

BRO 异常检测

- 案例4：通过bro脚本检测dns隧道

```
# DNS names to not alert on
const ignore_DNS_names = /wpad|isatap|autodiscover|gstatic\.com$|domains\._msdcs|mcafee\.com$/ &redef;
# size at which dns query domain name is considered interesting
const dns_query_oversize = 90 &redef;
```

判断dns请求是否大于设置的域名长度90，并且通过正则排除掉可能误报的情况

```
if (|query| > dns_query_oversize && ignore_DNS_names !in query)
        {
        NOTICE([$note=DNS::Oversized_Query,
                $conn=c,
                $msg=fmt("Query: %s", query),
                $sub=fmt("Query type: %s", qtype),
                $identifier=cat(c$id$orig_h,c$id$resp_h),
                $suppress_for=20min
                ]);
```

BRO 异常检测

- 案例4：通过bro脚本检测dns隧道
  bro –r dnscat2.pcap local –C bro读取pcap包进行解析
  cat notice.log 查看告警 type:5 指的是cname记录 txt对应16

```
ount    string  set[enum]       interval        bool    string  string  string  double  double
1473189242.582970       CiTdS32fcdH6sXv2Nd      192.168.198.137 38743   192.168.198.2   53      -       -       -       udp     DNS::
Oversized Query Query: 3f59038040000000000e4c55a46546fbf50d88a23817c278a3438b29b0cc.905cad5a3f7432cddcf86ebb11c965c6c3bde418bcc42edd4
d9670e00fef.766f33d5971818a4c5594fa9a9.1.eej.me Query type: 5   192.168.198.137 192.168.198.2   53      -       bro     Notice::ACTIO
N_LOG   1200.000000     F       -       -       -       -       -
#close  2018-06-04-09-04-38
```

BRO 异常检测

- 案例4：通过bro脚本检测dns隧道
  domain_stats.py 创建一个web服务，通过api接口确定一个域名是否在top-1m.csv列表中，白名单对比。

  echo 'p5-ammcgqbbyr3os-uvb3zkyjhbtdkh42-943555-i1-v6exp3.v4.metric.gstatic.com'|ag -o '[\w-]{2,15}\.\w{2,5}(?:\.[\w]{2,4})?$'|xargs -i{} curl http://127.0.0.1:8000/alexa/{}
  如果有对应的结果时则显示具体位置

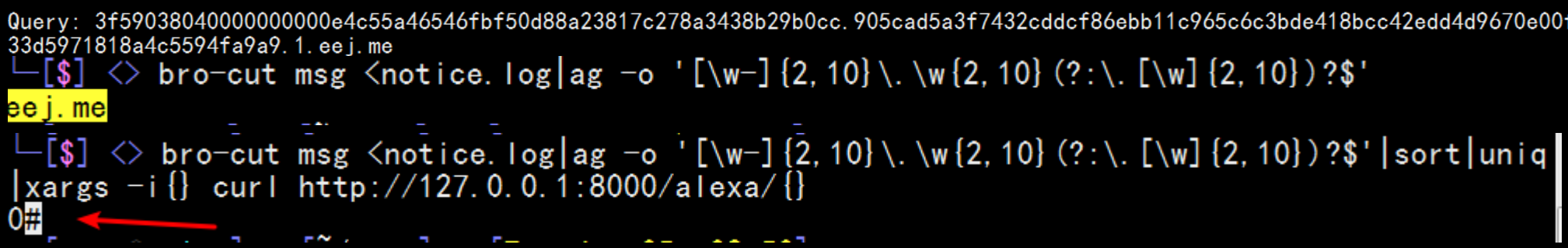
```
└─[$] <> echo 'p5-ammcgqbbyr3os-uvb3zkyjhbtdkh42-943555-i1-v6exp3.v4.metric.gstatic.com'|ag -o '
[\w-]{2,15}\.\w{2,5}(?:\.[\w]{2,4})?$'|sort|uniq|xargs -i{} curl http://127.0.0.1:8000/alexa/{}

1147#
```

BRO 异常检测

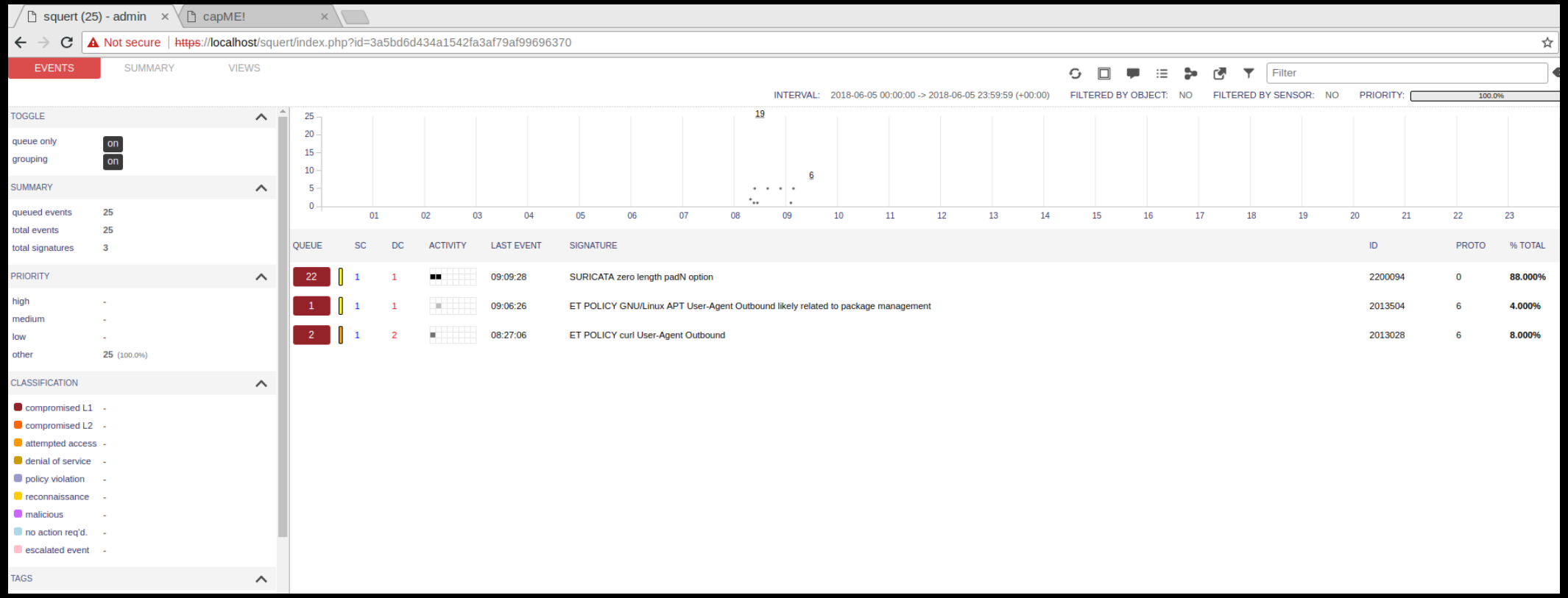
- 案例4：通过bro脚本检测dns隧道

bro-cut msg <notice.log|ag -o '[\w-]{2,15}\.\w{2,5}(?:\.[\w]{2,4})?$'|sort|uniq|xargs -i{} curl http://127.0.0.1:8000/alexa/{}



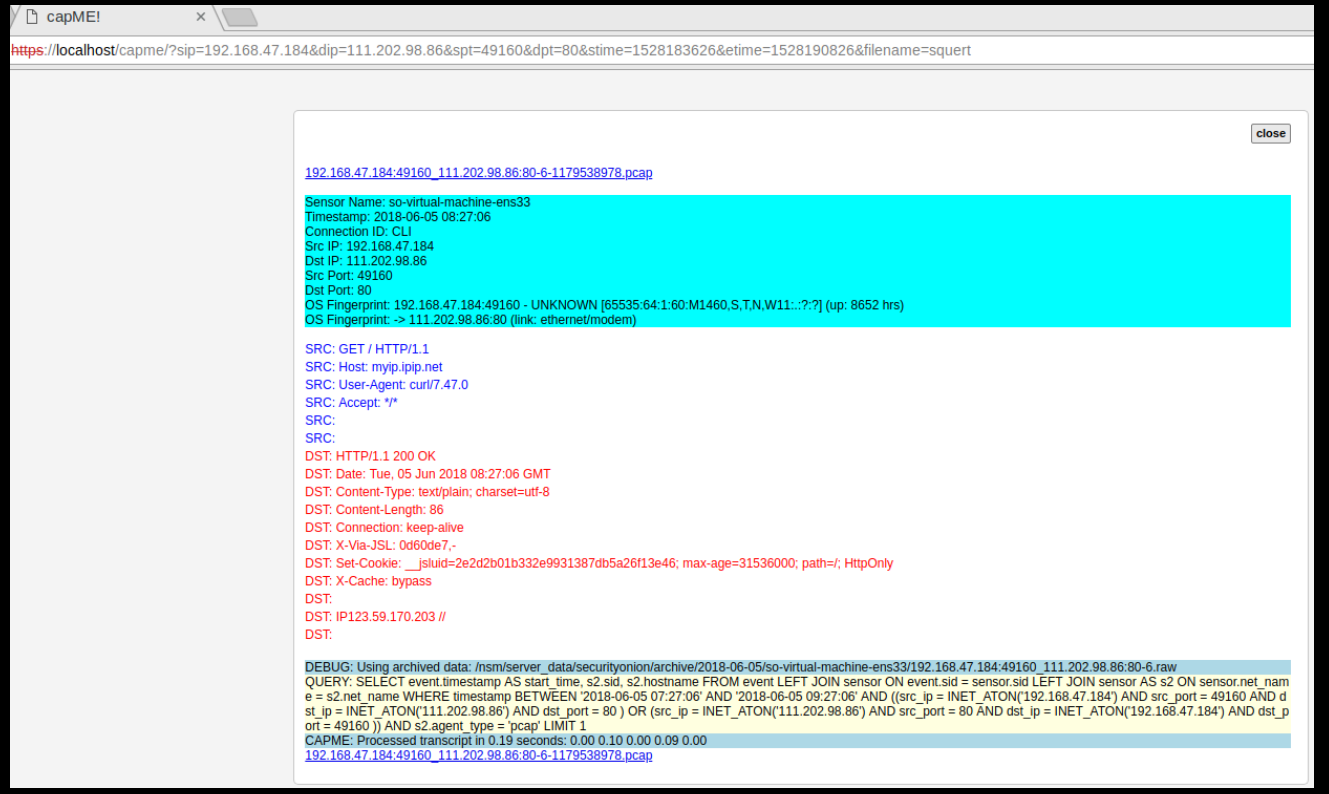
安全分析工具-Squert

- Squert是Sguil的图形界面，用于分析NIDS告警，规则id相同的告警都汇聚到1条信息里



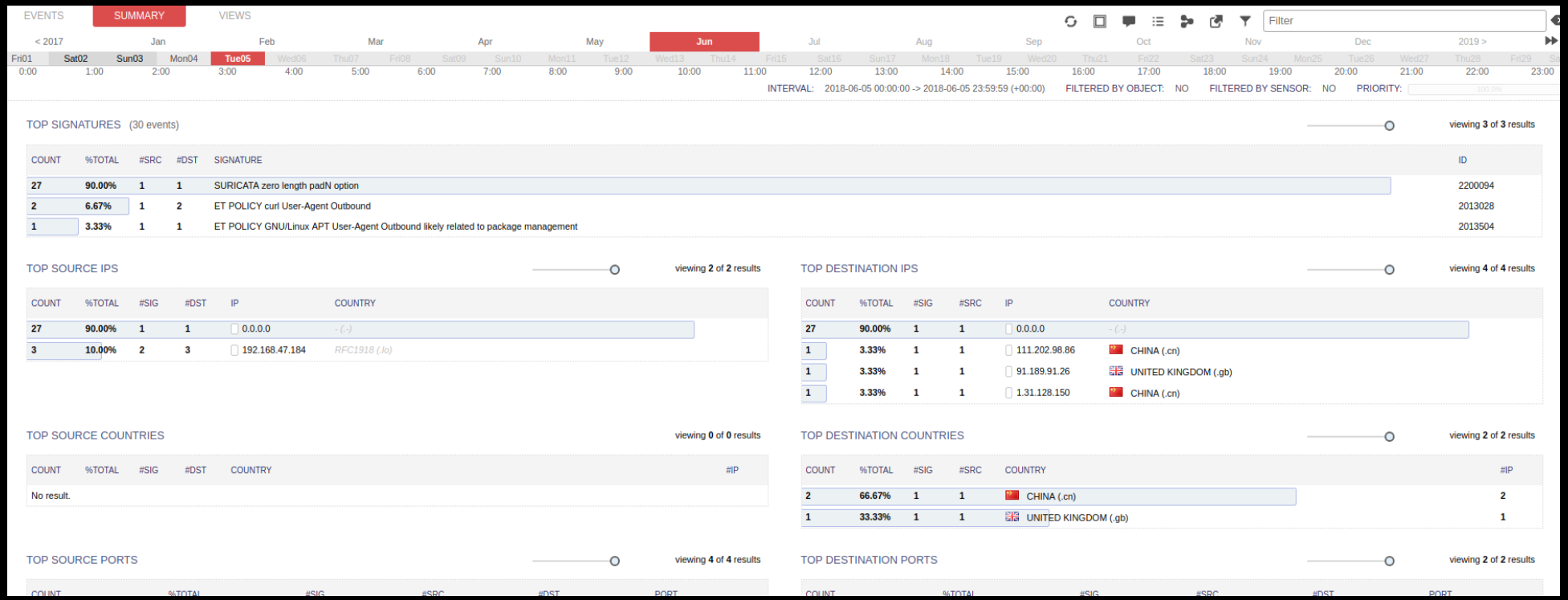
安全分析工具-Squert

- 在Squert的告警id里单击就会跳转到capme界面，capme会根据告警数据的源ip，源端口，目标ip，目标端口，时间在全流量数据里面去提取解码信息（包括解码gzip）。或单击pcap包下载wireshark分析



安全分析工具-Squert

- 点击SUMMARY，切换到Squert统计界面，会根据规则次数，源/目标ip，源/目标端口,国家等去统计



更多功能

- 使用ElastAlert 实现邮件告警功能
- 关联沙箱，通过bro提取文件然后输出到cuckoo（杜鹃）沙箱中，cuckoo运行后会给出一个评分1-10，风险低的将删除，风险高则告警。

参考来源

- https://github.com/Security-Onion-Solutions/security-onion/wiki
- 《网络安全监控：收集、检测和分析》
- 《网络安全监控实战 深入理解事件检测与响应》

关于我

- https://github.com/al0ne