Orchestrating Detection within Security Onion

#BlueTeamSummit 2020
@DefensiveDepth
Hi there,

On December 4, 2019, we identified a bug in our systems that affected how Cloud Firewalls were applied to some. From November 20, 2019 until December 5, 2019, Cloud Firewall rules were not enforced on some of your

We identified the root cause of the issue and reapplied all Cloud Firewalls. They are now operating normally.
“...self-contained, fully documented, prescriptive procedures for finding and responding to undesired activity.”

“By building the documentation and instructions into the play, we have directly coupled the motivation for the play, how to analyze it, the specific machine query for it...”
Hi there,

On December 4, 2019, we identified a bug in our systems that affected how Cloud Firewalls were applied to some. From November 20, 2019 until December 5, 2019, Cloud Firewall rules were not enforced on some of your

We identified the root cause of the issue and reapplied all Cloud Firewalls. They are now operating normally.
Motivation

What are we looking for and why?

This play focuses on finding Internet-exposed services that shouldn't be. These are services that were not initially configured to be Internet accessible, but are, whether through misconfiguration, system vulnerabilities, or maliciousness. Specifically, this play looks for Elasticsearch by using osquery to query for connections to the endpoint on TCP/9200 with a non-local remote address.

ATT&CK Mapping:

T1190 - Exploit Public-Facing Application
Machine Query

Actual search query needed

```
SELECT distinct process_open_sockets.local_address, 
    process_open_sockets.local_port, process_open_sockets.remote_address, 
    processes.path, processes.cmdline, '901' AS qid 
FROM process_open_sockets JOIN processes using (pid) 
WHERE process_open_sockets.local_port IN (3306, 1433, 5432, 27017, 6379, 5984, 9200) 
AND process_open_sockets.remote_address NOT LIKE '10.%' 
AND process_open_sockets.remote_address NOT LIKE '172.16%' 
AND process_open_sockets.remote_address NOT LIKE '192.168%' 
AND process_open_sockets.remote_address NOT LIKE '127.0.0.1' 
AND process_open_sockets.remote_address NOT LIKE '0.0.0.0' 
AND process_open_sockets.remote_address NOT LIKE '::' 
AND process_open_sockets.remote_address NOT LIKE '0';
```

(osquery.columns.qid:"901" AND 
osquery.columns.local_port:("9200"))
Next Steps

How to analyze the results

1) Manually confirm that the resource is accessible publicly
   eg. `curl es.defensivedepth.com:9200`

2) Contact system owner & start working on the next step

3) Review endpoint & network logs for unauthorized access
How Standards Proliferate:
(See: A/C chargers, character encodings, instant messaging, etc.)

**Situation:**
There are 14 competing standards.

**14?! Ridiculous!**
We need to develop one universal standard that covers everyone's use cases.

**Yeah!**

**Soon:**

**Situation:**
There are 15 competing standards.

xkcd.com/927
“Sigma is for log files what Snort is for network traffic and YARA is for files.”

Sigma Format
• Generic
• YAML

Sigma Converter
• Backend
• Custom field mapping

Machine Query
• Elasticsearch
• Splunk
• etc

github.com/Neo23x0/sigma
title: Internet Exposed Services - ElasticSearch

description:
This play focuses on finding Internet-exposed services that shouldn’t be. These are services that were not configured to be Internet accessible, but are, whether through misconfiguration, system vulnerabilities, or maliciousness. Specifically, this play looks for Elasticsearch by using osquery to query for connections to the endpoint on TCP/9200 with a non-local remote address.

status: experimental
author: Josh Brower
tags:
- attack.initial_access
- attack.t1190

logsource:
product: osquery
osquery_qid: 901
osquery_logging: differential
osquery_table: process_open_sockets

falsepositives:
- Elasticsearch instances that are meant to be publically accessible via port 9200. This is not common.

level: high
tasks:
1 - Confirm findings: Manually confirm that the resource is accessible publicly – "curl <IP>:9200"
2 - Contact system owner: Contact system owner and start working on the next step (reviewing logs for unauthorized access)
3 - Review logs: Review endpoint & network logs for unauthorized access
(osquery.columns.qid:"901" AND osquery.columns.local_port:"9200")
Abstract to Concrete

How to do this in Security Onion?
DEMO
Orchestrating Detection within Security Onion

#BlueTeamSummit 2020
@DefensiveDepth