LINUX INVESTIGATIONS AT SCALE

SANS DFIR Summit September 2019

John Rogers & Joani Green
INTRODUCTION

JOHN ROGERS - WHOAMI
- Senior Incident Response Consultant, F-Secure Consulting JHB
- Strategic Cyber Defense Projects
- Tinkering with electronics and home projects

JOANI GREEN - WHOAMI
- Senior Incident Response Consultant, F-Secure Consulting London
- OSCP, Purple Teamer, Forensics, IR
- #WoSEC #WomenThinkCode

WoSEC - Women of Security
@WoSECtweets

WomenThinkCode - @WomenThinkCode
WHY LINUX FORENSICS?
THE PROBLEMS THAT LED TO THIS PROJECT

⚠️ Limited ability to triage and investigate Linux systems at scale
⚠️ Too much raw data to analyse
⚠️ Limited technology
CASE STUDIES

CASE STUDY 1:

- Entire Windows estate compromised
- No visibility of Linux environment, limited logs and central log aggregation, no Linux EDR
- Corporate web servers + client data
- Administrator → Y0u11n3v3rgu3ss! ← Root
CASE STUDIES

CASE STUDY 2:

- Mass exploitation of a new vulnerability
- Limited time to perform IR against servers from different clients
- Example: Drupalgeddon or new SSH RCE
CASE STUDIES

CASE STUDY 3:

- "Compromise Assessment"
- historical compromise
- ~180 Linux endpoints to analyse
FORENSICS - METHODOLOGY

- Always start with what you know
- Baseline data
- Timeline analysis

- find one or two suspicious files/entries and expand your investigation from there

LOCARD’S EXCHANGE PRINCIPLE:
"The perpetrator of a crime will bring something into the crime scene and leave with something from it, both elements can be used as forensic evidence.”

- Find at least 1 indicator that an attacker brought or took something
How reproducible is the Windows forensics process to the Linux OS?

A quick comparison ...

- **MFT**
  - **Windows**
  - **Linux**: Metadata / Inodes

- **Windows Event Logs**
  - **Windows**
  - **Linux**: /var/log/*

- **Registry**
  - **Windows**
  - **Linux**: Everything is a file
**NETWORK**
- Processes communicating to strange ports
- Uncommon IP addresses (Deviation from baseline)
- Known-bad IP address communication
- Endpoints communicating to each other (Deviation from baseline)

**LOGS**
- `\var\log\*`
- Enumeration and brute forcing attempts should be obvious in logs
- Auth, last, btmp, utmp, wtmp, syslog
- Search for log file tampering

**PROCESSES**
- Strangely named processes, look legit but have small deviations
- Processes communicating to IP addresses
- High CPU/RAM
- Deleted binaries that are still running
- Known bad hash matching
LINUX FORENSICS FUNDAMENTALS

- Newly created users
- Recent permission changes
- Recent logins from known suspicious users
- Bash History – weird commands and clearing
- Suspicious SSH keys, for example does wwwdata user have an authorized key?
- Persistence mechanisms via chron etc.

FILES
- Exploit traces
- Immutable files
- File Masquerading
- Hidden Binaries

- Named pipes
- System files modified
- Known-bad files present

USERS

- Creating a timeline is fundamental to building the bigger picture
- Identify 1 IoC and follow the story
LINUX FORENSICS — IMPORTANT ARTEFACTS

**LIVE DATA**
- Network communication (netstat)
- Currently logged in users
- Running processes
- Resources

**HISTORICAL DATA**
- Bash history (sometimes timestamped)
- Recently logged in users
- Login failures
- Authorised keys
- Binaries
- Persistence mechanisms
- /var/log/*
INITIAL APPROACH:
▪ Bash script to collect forensic data from remote endpoints

CHALLENGES FACED:
▪ 3rd party binaries (log2timeline etc)
▪ Disclose sensitive information
▪ Performance and Disk usage
▪ Too much data to do disk or live forensics at scale

SOLUTION:
▪ Bash script which relies on native Linux commands
PHASE 1: COLLECTION

HOW WE APPROACHED THIS:
- Provide collection script

CHALLENGES FACED:
- Manual deployment and collection

SOLUTION:
- Deploying scripts to endpoints at scale and executing them e.g. Puppet
- Push results to dedicated server for collection
PHASE 2: PARSING DATA

HOW WE APPROACHED THIS:
- grep/awk/sed

CHALLENGES FACED:
- Time intensive, boring, tedious work

SOLUTION:
- Ingest the data into an ELK stack for easy analysis.
PHASE 2: PARSING DATA
WHAT THE ELK?

Elastic Search, Logstash, Kibana

Large open source community with grok filters for common log formats

- Dashboard support for fancy visualisations
- Easy to customise data fields

Can readily be created using Docker

https://github.com/deviantony/docker-elk/
https://github.com/thomaspatzke/logstash-linux
https://github.com/philhagen/sof-elk/ [FOR 572]
PHASE 2: PARSING DATA
WHAT THE ELK?

Kibana allows graphical representation of data making it easier to spot anomalies and reduce the number of false positives

Grok filters can be modified to add new fields to the data to simplify the statistical analysis process

https://grokdebug.herokuapp.com/

https://grokconstructor.appspot.com/
PHASE 2: PARSING DATA
WHAT THE ELK?

```json
input {
  file {
    path => [ "/logs/**-netstat.txt" ]
    start_position => beginning
  }
}
filter {
  grok {
    match => { "path" => "%{GREEDYDATA:Hostname}- %{YEAR:Year} %{MONTHNUM:Month} %{MONTHDAY:Day}-%{HOUR:Hour} " }.
    add_field => [ "datetime", %{Day} %{Month} %{Year} %{Hour}: %{Minute}"
    match => { "message" => "%{MAIN}" }
  }
}
output {
  elasticsearch {
    hosts => "elasticsearch:9200"
    index => "snap-netstat"
  }
}
```
<table>
<thead>
<tr>
<th>Time</th>
<th>Message</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aug 19, 2019 05:25:01.000</td>
<td>pan_unix(cron:session): session opened for user root by (uid=0)</td>
<td>varlog-auth</td>
</tr>
<tr>
<td>Aug 19, 2019 05:25:01.000</td>
<td>pan_unix(cron:session): session closed for user root</td>
<td>varlog-auth</td>
</tr>
<tr>
<td>Aug 19, 2019 05:35:01.000</td>
<td>pan_unix(cron:session): session opened for user root by (uid=0)</td>
<td>varlog-auth</td>
</tr>
<tr>
<td>Aug 19, 2019 05:35:01.000</td>
<td>pan_unix(cron:session): session closed for user root</td>
<td>varlog-auth</td>
</tr>
<tr>
<td>Aug 19, 2019 05:39:01.000</td>
<td>pan_unix(cron:session): session opened for user root by (uid=0)</td>
<td>varlog-auth</td>
</tr>
<tr>
<td>Aug 19, 2019 05:39:01.000</td>
<td>pan_unix(cron:session): session closed for user root</td>
<td>varlog-auth</td>
</tr>
<tr>
<td>Aug 19, 2019 05:43:01.000</td>
<td>pan_unix(cron:session): session opened for user root by (uid=0)</td>
<td>varlog-auth</td>
</tr>
<tr>
<td>Aug 19, 2019 05:43:01.000</td>
<td>pan_unix(cron:session): session closed for user root</td>
<td>varlog-auth</td>
</tr>
</tbody>
</table>
PHASE 3: ANALYSIS

HOW WE APPROACHED THIS:

▪ Large amounts of data, limited IoCs
▪ Looking for known bad - run hashes through virus total, network analysis, manual analysis of bash history, persistence locations.

CHALLENGES FACED:

▪ Tedious and time consuming,
▪ Historical or point-in-time data doesn't always reveal the bad stuff
PHASE 3: ANALYSIS

SOLUTION:

▪ Detection
  Python & Bash to parse the data using Linux compromise detection rules (Sigma project)

▪ Isolating the juicy stuff
  Pulling out the data that will most likely reveal any IoCs for easier analysis in the ELK stack. This is also the data that can be given to the client to differentiate normal from abnormal

▪ Reporting
  Automating the reporting process with Python and Sharelatex
PHASE 3: ANALYSIS

- [https://github.com/Neo23x0/sigma](https://github.com/Neo23x0/sigma)

**Sigma**

Generic Signature Format for SIEM Systems

**What is Sigma**

Sigma is a generic and open signature format that allows you to describe relevant log events in a straightforward manner. The rule format is very flexible, easy to write and applicable to any type of log file. The main purpose of this project is to provide a structured form in which researchers or analysts can describe their once developed detection methods and make them shareable with others.

Sigma is for log files what Snort is for network traffic and YARA is for files.

This repository contains:

- Sigma rule specification in the *Wiki*
- Open repository for sigma signatures in the `rules` subfolder
- A converter that generates searches/queries for different SIEM systems (work in progress)

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>Fixes for Elasticsearch query correctness CI tests</td>
</tr>
<tr>
<td>apt</td>
<td>rule: emissary panda activity</td>
</tr>
<tr>
<td>compliance</td>
<td>Added level</td>
</tr>
<tr>
<td>linux</td>
<td>fix: linux cmds rule</td>
</tr>
<tr>
<td>network</td>
<td>Merge pull request #315 from P4T121CK/feature/net_dnc_c2_detection</td>
</tr>
<tr>
<td>proxy</td>
<td>docs: renamed file name</td>
</tr>
<tr>
<td>web</td>
<td>Web Source Code Enumeration via .git</td>
</tr>
<tr>
<td>windows</td>
<td>fix: renamed powershell rule</td>
</tr>
</tbody>
</table>
PHASE 4: HUNTING

- A human analyst is still needed
- Frequency analysis of processes/account usage etc.
- Netstat/SSH visual map
- Dashboards and other general observations
- With new findings, add to the ruleset to enhance automation ability
NETSTAT VISUALIZATION:
SUMMARY OF THE PROJECT

- Collection Script (Bash script)
- Collection Output
- Extractor (Bash script)
- ELK Stack Docker image
- Auto Analyzer & Report Generator (Bash-flavoured Python script)
FUTURE WORK

- Optimise scripts
- Customised Dashboards
- SFTP Integration + Ingestion
SUMMARY OF THE PROJECT

- https://github.com/FSecureLabs
LESSONS LEARNT

▪ You don’t have to be a supreme technical genius to solve a problem
▪ There are plenty of experts who are usually willing to help
▪ You don’t have to be a developer to script
▪ Automating the boring stuff = happiness

▪ Shout out to our awesome colleagues
QUESTIONS?