Unconventional Logging and Detection

Justin Henderson (@SecurityMapper)
About Me

Justin Henderson
- SEC555 Author / SEC455 and SEC530 Co-Author
- GSE #108 / Cyber Guardian Blue + Red / 60 certs
- Owner and Principal Consultant of H & A Security Solutions
- Twitter: @SecurityMapper
- GitHub: github.com/HASecuritySolutions
- And security hobbyist and community supporter
  - Collecting interns/contributors in bulk (research teams)
  - Release research to the community
Unconventional Tactics

Log Collection

- Focus on picking up logs
  - Without requiring agents on each endpoint
  - Or without agents entirely
- Upgrading logs
  - Adding context (log enrichment)

Methods work regardless of SIEM in use

Detection

Implement detection capabilities

- Without requiring security software or agents
- Remove requirement for remote polling software
  - osquery
  - SSH
  - WMI
Log Agents

Remote Site

agent

agent

agent

syslog

Internet

VPN Tunnel

Central Location

Storage

Log Aggregator
Windows Event Forwarding

To collect logs using Windows event forwarding is a two-step process

• An event collector must be set up
• Then, either the collector must be configured to pull events or endpoints must be configured to push events

GPO is used to tell endpoints what logs to push
• Configuration of event selection is similar for push/pull

The end destination is intended to be Windows, not SIEM
Blind Drop

Windows event forwarding cannot ship file logs
• Only handles native Windows events

Business requirement may require no third-party agents
• Can be handled by using a blind file share

Requires a single file server to have a third-party agent or PowerShell script
• Share is created with write permissions only
• Log files are then saved to this share
Blind Drop with PowerShell Transcription Logging

List folder contents at root folder, write only on subfolders
Log Collection of Key Data Source != Good Data

A SIEM is often treated as a mainstream consumer of data

- Yet it should emphasize analysis and detection

**Case Study / Example** - Which would you rather analyze?

<table>
<thead>
<tr>
<th>AppLocker Block Event</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File</strong> = PATH/SEC555_ROCKS.EXE</td>
</tr>
<tr>
<td><strong>File Hash</strong> = empty</td>
</tr>
<tr>
<td><strong>User</strong> = jhenderson</td>
</tr>
<tr>
<td><strong>Policy Name</strong> = exe</td>
</tr>
</tbody>
</table>

%OSDRIVE%\SEC555_ROCKS.exe was prevented from running.

<table>
<thead>
<tr>
<th>Modified Block Event</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File</strong> = PATH/SEC555_ROCKS.EXE</td>
</tr>
<tr>
<td><strong>File Hash</strong> = empty</td>
</tr>
<tr>
<td><strong>User</strong> = jhenderson</td>
</tr>
<tr>
<td><strong>Policy Name</strong> = exe</td>
</tr>
<tr>
<td><strong>Signed</strong> = false</td>
</tr>
<tr>
<td><strong>Signature</strong> = Unknown</td>
</tr>
<tr>
<td><strong>Hashes</strong> = MD5</td>
</tr>
<tr>
<td><strong>Sandbox</strong> = 9/10 evilness rating</td>
</tr>
</tbody>
</table>
Problem is how to gain context with a SIEM (multiple options)

1. Log enrichment during **data ingestion** (Example - Logstash)
2. Log enrichment while **viewing the log** (Example - Splunk)
3. Log enrichment via **orchestration** (Example - Phantom)

Above implementations are tool specific (use of all would be ideal)

- And may be difficult to maintain / implement
- Any certain techniques do not scale well with above methods
  - Waiting for malware sandbox analysis takes minutes
Different Approach

1. Create a Basic Task
2. Task Trigger
3. When a Specific Event Is Logged
4. Action
5. Program/script: `powershell.exe`

- Add arguments (optional): `-file script_goes_here`
- Start in (optional):
When task schedule is triggered the script could do the following:

- Get the recent 8004 event ID from AppLocker
- Send the file to malware sandbox
- Run `sigcheck.exe` against file
- See if `Sysmon` events exist to find parent process info
- Perform any other additional checks
- Write a new custom log
Custom Logging with PowerShell

PowerShell makes writing custom logs easy

• Simply use Write-EventLog to generate a custom log
• If you want a custom channel, first, use New-EventLog

1. `New-EventLog -LogName LogCampaign -Source CustomLogs`
2. `Write-EventLog -LogName LogCampaign -Source CustomLogs -EventId 31337 -EntryType Warning -Message "New and improved log goes here"

Can be used to generate logs that even basic log collectors can collect

• May remove the need for special log agents
PowerShell Generated Log
Offloading Job

Standard user privileges work with writing logs to Windows
• But custom channel must be created first
• Does not work to pre-built channels like Security
• Crafting log over network works as standard user
Possible to offload sensitive jobs off endpoints
• Example - Trigger custom log and copy file to blind share
  • Then perform automated analysis there
• Use SIEM to trigger automatic action
Custom Logging via Scripts

**PowerShell**

1. Install-Module Posh-Syslog
2. Import-Module Posh-Syslog
3. Send-SyslogMessage -Server "x.x.x.x" -Message 'Log message here' -Severity Informational -Facility syslog -Transport TCP

**Python**

```python
1 import logging
2 import logging.handlers
3 custom_logger = logging.getLogger('MyLogger')
4 custom_logger.setLevel(logging.INFO)
5 handler = logging.handlers.SysLogHandler((address = ('X.X.X.X', 514))
6 custom_logger.addHandler(handler)
7 custom_logger.info("Log message here")
```
Custom Format

Scripting log allows output to be in any format

- CEF
- LEEF
- Syslog
- JSON
- Key-value
- CSV
Unconventional Logging

Collection

Multiple options outside of using agents to collect logs

• Script out / to Windows events
• Use script to send logs via syslog or custom formats
• Output as file to central file share
• Use Windows Event Forwarding

Enrichment

Take existing logs and make them better

• Windows Task Scheduler
  • To kick off script based on Windows Event ID
  • Or read log with script
Then generate better log(s)
* Still need aggregator, search, and SOAR for other enrichment
Agentless Detection

Adversaries "live off the land" and use tools against you

- But defenders need to and should "live off the land"
- **PowerShell, Python, and Bash** provide scalable, easy to implement detection and response capabilities

Try running simple scripts on a re-occurring basis

- **Windows Task Scheduler** running every 5 minutes
- **Cron job** running every 1 minute
- Run at **startup script** that handles its own scheduling
Log Campaign - GitHub Project

Provides detection capabilities with logging Campaigns

File Ingestion --->
Provides ability to read in files, convert them, and log to syslog or evtx
Modular Campaigns

Each Campaign is its own module
• Allows easy drop in of custom modules
• Or using pre-built modules

Example - **ARPCache.ps1** module
• On first run, saves ARP entry for default gateway
• Then generates log if MAC address changes
• Acts as an alert to ARP cache poisoning attacks
Campaigns can call 3\textsuperscript{rd} party binaries like \texttt{autorunsc.exe}

- Many organizations use autorunsc.exe for incident handling / forensics for finding persistence
- Works well for local or file server storage but not daily baseline to SIEM (or near real time monitoring)

\texttt{Autoruns.ps1} module calls \texttt{autorunsc.exe}

- Generates logs only when new or changed entries found
- Can be run every X minutes without your SIEM dying
Baseline Data Sources

Good data requires getting your hands dirty
• May have to go "fetch" what you need

Useful data
• Active processes
• Certificates
• Drivers
• Host files
• Registry keys
• Route table
• Scheduled tasks
• Security status
• Services
• Shares
• Software
• USB devices
• Users and groups
Log Campaign supports multiple methods of scheduling:

- Control schedule via **Windows Task Scheduler**
- Implement schedule using **-Schedule** parameter:
  - Supports per X minute repetitive scheduling
  - Great when combined with a startup task or script

```
./log_campaign.ps1 -Campaign all -Schedule 1
```

Run all **Campaigns** or specific ones on custom schedules.
Local Detection Strategies

- **Matching** (Blacklist) - Look for something that should not occur then generate alert
- **Anomaly** - Look for things that change
  - To save time and resources consider only sending "new" items
- **Change Monitoring** - Look for things that change
  - Treat as alert if change is not authorized / expected

Local use can be combined with things like DeepBlueCLI

- Any tool that can output to file may be easier to maintain
GitHub Link

Log Campaign can be found in this GitHub repo:
https://github.com/HASecuritySolutions/LogCampaign

Feel free to modify or make a Python or Bash equivalent