Enabling the Detection Development Lifecycle with Attack Simulation

Augusto Barros
VP Security Evangelist

Volkan Ertürk
CTO & Co-Founder
Agenda

- The Detection Development Life-Cycle
- How can Attack Simulation Support the DDLC?
Detection Development?

- Many threat detection tools have separate “content” that defines what they should be looking for
- A.K.A “use cases” (SIEM)
- Can also include signatures, policies and rules for tools such as EDR, NDR and DLP
- Content can be provided by the vendor or developed by the user
  - Is the content provided by the vendor enough, or aligned to what you need?
SIEM Use Case

● As described by Gartner:
  “a specific set of conditions or events, usually related to a specific threat, to be detected or reported by the security tool”

● Usually seem as (a) rule(s)
  ○ More complex scenarios require multiple rules, ML models, threat chains and other resources
The Detection Development Life Cycle

A mature threat detection practice must include processes to identify, develop, implement and refine detection content.
Why a “development life-cycle”? 

- Organizations most successful with threat detection content adopt a code development approach to it
- But cannot be too complex
  - Agility is important
  - An Agile style approach is ideal
Where to start?

Major problem with use cases is answering the question:

Which ones do I want to deploy on my SIEM?
Identifying Use Cases

You need to know first what you want to accomplish

One day Alice came to a fork in the road and saw a Cheshire cat in a tree. "Which road do I take?" she asked. "Where do you want to go?" was his response. "I don't know," Alice answered. "Then," said the cat, "it doesn't matter."
### Prioritizing importance

<table>
<thead>
<tr>
<th>#</th>
<th>PICUS</th>
<th>CrowdStrike</th>
<th>Recorded Future</th>
<th>redcanary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Process Injection</td>
<td>Masquerading</td>
<td>Security Software Discovery</td>
<td>Process Injection</td>
</tr>
<tr>
<td>2</td>
<td>PowerShell</td>
<td>Command-line Interface</td>
<td>Obfuscated Files or Information</td>
<td>Scheduled Task</td>
</tr>
<tr>
<td>3</td>
<td>Credential Dumping</td>
<td>Credential Dumping</td>
<td>Process Injection</td>
<td>Windows Admin Shares</td>
</tr>
<tr>
<td>4</td>
<td>Masquerading</td>
<td>PowerShell</td>
<td>System Information Discovery</td>
<td>PowerShell</td>
</tr>
</tbody>
</table>

Source: Dr. Suleyman Ozarslan (@su13ym4n)
Prioritizing Feasibility

- Do you have the logs?
- Do you have context data?
- Do you have the tools?
- Can you handle the output?
  - People (# of analysts)
  - Technology (capacity)
  - Process (playbooks)
Implementation

- “Detection as Code” is becoming popular
- Strong analytics capabilities help increasing implementation options
  - What doesn’t work as a rule may work as ML model
- Out of the box content can speed up implementation of initial use cases
- Community efforts to share content as growing (e.g. Sigma)
Testing

- Testing SIEM content is not always easy
  - Do you have a test environment?
  - Do you have the right data in the test environment?

- Testing environments are costly and it's hardly to replicate production context
  - User activity + Attacks

- Attack simulation allows testing in the production environment
Refining

The good and old “tuning”

- Do not believe in who tells you “there is no tuning”
- It’s not only about False Positives
- Prevalence of events matter!
- Can you answer “Is it still working?”
Measuring the Process

- A DDLC Process provides useful metrics
  - Metrics from unfeasible use cases
    - Data sources missing
    - Tools deficiencies
  - Metrics from tuning use cases
    - Implementation quality
    - Tools deficiencies
    - Data quality issues
  - Metrics from attack simulation
    - Gaps in coverage
    - Efficiency problems
How Can Attack Simulation Support the DDLC?

- Developing detections can be overwhelming
  - What to do?
  - What to do first?
  - Is it working?
- Attack Simulation can help in all these challenges
We have a detection development methodology!

Yet successful implementation is challenging.

- Manual and time-consuming
- Requires diverse skill-set
- Error-prone
- Ever-changing threats
How to Tackle

Empower Detection Development Lifecycle with Attack Simulation

1. Threat Selection
2. Adversary Emulation
3. Log Validation
4. Alert Validation
5. Continuous Improvement
1. **Threat Selection**

   Identify the relevant set of threats to validate the use-case.

   - Available threat content
   - Based on your past incidents
   - Threat Intelligence
1. Threat Selection

Identify the relevant set of threats to validate the use-case.

- Start with “Relevant Threats” that you do not have visibility of or based on your past incidents
- Start with “your weakest technique” or “most used”
1. Threat Selection

Prioritization Idea:

Among the shortlisted threats, look for quick wins (telemetry data availability).
2. Adversary Emulation

Create an adversary emulation plan and execute.

Adversary Emulation Process recommended by Mitre ATT&CK

<table>
<thead>
<tr>
<th>Category</th>
<th>Built-in Windows Command</th>
<th>Cobalt Strike</th>
<th>Metasploit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1062</td>
<td>net config workstation</td>
<td>net config</td>
<td></td>
<td>Get the Windows OS version that’s running.</td>
</tr>
<tr>
<td>T1062</td>
<td>net config workstation</td>
<td>net config</td>
<td></td>
<td>Get element of environment variables.</td>
</tr>
<tr>
<td>T1033</td>
<td>whoami /all /fo list</td>
<td>whoami /all /fo list</td>
<td></td>
<td>Get the user is who the user belongs to, security permissions of the user.</td>
</tr>
<tr>
<td>T1092</td>
<td>net config server</td>
<td>net config server</td>
<td></td>
<td>Get computer name, usernames, OS software version, domain information, DNS, logon domain.</td>
</tr>
<tr>
<td>T1016</td>
<td>pconfig get</td>
<td>pconfig get</td>
<td></td>
<td>Get information about the domain, network adapter ([DNS]/[WINS] server).</td>
</tr>
<tr>
<td>T1092</td>
<td>systeminfo (i.e. COMPNAME) [u DOMAIr髄User [p password]]</td>
<td>systeminfo (i.e. COMPNAME) [u DOMAIr髄User [p password]]</td>
<td>systeminfo if you already have a systeminfo run winenum.get_env.rb</td>
<td>Get information about the system.</td>
</tr>
<tr>
<td>T1011</td>
<td>reg query &quot;HKLM_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\TerminalServer%\r\n\r\nNetServerList&quot;</td>
<td>reg query &quot;HKLM_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\TerminalServer%\r\n\r\nNetServerList&quot;</td>
<td>reg query [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\TerminalServer%\r\n\r\nNetServerList&quot;</td>
<td>Check the current registry value for terminal services, if it’s 0, then terminal services are enabled, if it’s 1, then they’re disabled.</td>
</tr>
<tr>
<td>T1016</td>
<td>arp -a -route print</td>
<td>arp -a -route</td>
<td>route print</td>
<td>Display the ARP table.</td>
</tr>
<tr>
<td>T1049</td>
<td>netstat -ano</td>
<td>netstat -ano</td>
<td></td>
<td>Displays list of currently running processes and services on the system.</td>
</tr>
<tr>
<td>T1057</td>
<td>tasklist /T [login]</td>
<td>tasklist /T [login]</td>
<td></td>
<td>Display list of currently running processes and services on the system.</td>
</tr>
<tr>
<td>T1099</td>
<td>net localgroup &quot;Administrators&quot;</td>
<td>net localgroup &quot;Administrators&quot;</td>
<td></td>
<td>Display the list of domain administrator accounts.</td>
</tr>
<tr>
<td>T1069</td>
<td>net group [&quot;Domain Admins&quot;] [Domain]</td>
<td>net group [&quot;Domain Admins&quot;] [Domain]</td>
<td></td>
<td>Display the list of domain administrator accounts.</td>
</tr>
<tr>
<td>T1087</td>
<td>net user [username] [domain]</td>
<td>net user [username] [domain]</td>
<td></td>
<td>Display the list of domain administrator accounts.</td>
</tr>
<tr>
<td>T1018</td>
<td>net group [&quot;Domain Computers&quot;] [Domain]</td>
<td>net group [&quot;Domain Computers&quot;] [Domain]</td>
<td></td>
<td>Display the list of domain computers in the domain by showing their computer accounts (COMPUTER NAMES).</td>
</tr>
</tbody>
</table>
2. Adversary Emulation

Leveraging 1000+ adversary emulation plans within Picus Threat Library
3. Log Validation

What We Need

- Right logs at the right verbose level
- Proper parsing and storage by the SIEM

How to Validate

- Define expected data-sources
- Identify logs from expected data-sources
- Check required logs against simulated attacks
3. Log Validation

Picus Threat Emulation Solution

- Attack
- IPS
- FW
- PROXY

Attack Simulation
- IPS
- FW
- PROXY

Picus Manager

SIEM Solutions
- Splunk
- Elastic
- IBM QRadar
- McAfee ESM
- Micro Focus ArcSight
- Microsoft Azure Sentinel

EDR Solutions
- VMWare Carbon Black
- Microsoft Defender for Endpoint
3. Log Validation

227271 - CreateRemoteThread Process Injection with COM by using DynamicWrapperEx

Detection Result: Detected
Alert Result: Not Alerted
Blocking Result: Not Blocked

-00:00:09
Attack Started
27.05.2021 - 23:03:48

-00:00:05
Attack Ended
27.05.2021 - 23:03:56

-00:00:00
Attack Logged
27.05.2021 - 23:03:52
Elasticsearch

No Alerts
4. Alert Validation

What We Need

- Right rules to manage the alert fatigue
- Being up to date for new TTPs of that threat

How to Validate

- Query alerts for the simulation agents
- Check required alerts against simulated attacks
4. Alert Validation

449381 - Gootkit Banking Malware Attack Scenario

Detection Result: Detected
Alert Result: Alerted
Blocking Result: Blocked

- Attack Started: 23.11.2021 - 15:06:33
- Attack Ended: 23.11.2021 - 15:06:45
- Attack Logged: 23.11.2021 - 15:11:44 Elasticsearch
5. Continuous Improvement

Challenges

- Configuration drift
- Ever-changing threat landscape
- Managing the complexity of security tools
- Communication problems between the involved parties

How to improve

- Leverage automation opportunities for use-case development
- Streamline detection development process (CI/CD)
- Challenging ourselves against new threats
5. Continuous Improvement

Monitor your threat coverage against sudden drops and improvements.
SOC teams may leverage 3rd party contents to accelerate their detection content development process.

Lots of alert/correlation rule libraries available:
- Repository of your SIEM/EDR vendor
- Open-source libraries/projects
- Content libraries (ie Picus Mitigation Library)

Either we implement them directly or use them as templates to craft your detection rule, these rules need to be validated too!
Picus Detection Content Library

800+ detection rules (+log source recommendations), mapped to TTP’s and ATT&CK category mappings.

<table>
<thead>
<tr>
<th>Rule Id</th>
<th>Rule Name</th>
<th>Severity</th>
<th>Release Date</th>
<th>Update Date</th>
<th>MITRE ATT&amp;CK</th>
<th>Action Name</th>
</tr>
</thead>
</table>
| 3918    | Process Termination via PowerShell | Medium   | 01-09-2020   | 04-11-2021   | Impact                | Kill Specific Processes using Powershell ...
Terminates Specific Process via Powershell |
| 6105    | Execution of Encoded String or Command via... | Medium   | 14-09-2020   | 04-11-2021   | Defense Evasion       | Execute Encoded PowerShell Command ...
Execute Encoded PowerShell Command ...
Execute Powershell Script by using VBA ...
Execute Shellcode In Winword.exe Proc...
| 5104    | Persistence via File Transport to Word Startu... | Low      | 14-10-2021   | 14-10-2021   | Persistence           | Copy a File "winlog.wif" in MS Word Star... |

### Table Columns Explained
- **Rule Id**: Unique identifier for each detection rule.
- **Rule Name**: Description of the detection rule.
- **Severity**: Level of threat, ranging from Low to Critical.
- **Release Date**: Date when the rule was first published.
- **Update Date**: Date of last update to the rule.
- **MITRE ATT&CK**: Categories and tactics from the MITRE ATT&CK framework.
- **Action Name**: Actions that can be taken in response to the detection rule.
Detection Development Life-Cycle
Leveraging Attack Simulation

**Threat Selection**
Mobilize TTPs relevant to your environment with a few clicks in minutes.

**Attack Simulation**
Run attack simulations against your network, endpoint, and cloud security controls.

**Log/Alert Validation**
Identify your alerting gaps automatically.

**Rule Development**
Get actionable guidance to fix your alerting problems.

Continuous Improvement
Summary

- Detection content is critical to detection success
- Detection content creation and management requires a structured process
- Attack Simulation enables the DDLC by supporting multiple phases of the process, from identification to measurement
- Pre-built detection content and tests accelerates time-to-value and reduces implementation costs
THANK YOU!