**PROACTIVE DETECTIONS PROCESS** - We are aiming to move from ad-hoc querying to a tested repeatable, measurable process for detecting threats and plugging gaps in visibility in our EDR solution - Microsoft Defender for Endpoint

**1. Introduction: The opportunity with M365 Security Portal for proactive investigations**

Establishing a robust **detection process** is crucial for proactively mitigating cyber threats.

**2. Use a standard process for building, testing, deploying detection code**

We will follow the standard process as outlined in **Detection Engineering Maturity Models (DEMMs)**.

These models provide a framework for visualizing the progression of a detection program, outlining stages from initial, unstructured efforts to highly optimized and automated capabilities.

A common theme across these models is a**gradual evolution from ad-hoc practices to more systematic and ultimately optimized approaches.**

Recognizing the different stages allows a team to identify their current position and set realistic and achievable goals for future development.

To base our efforts on a systematic approach, we will rely on the Elastic Detection Engineering Behavior Maturity Model : <https://www.elastic.co/security-labs/elastic-releases-debmm>

| **Threat Hunting Maturity Model for MVG** | | |
| --- | --- | --- |
| **Feature of Detection Process** | **DEBMM Tier 0: Foundation** | **DEBMM Tier 1: Basic** |
| Rule Development & Management | Ad-hoc, unstructured, untested | Systematic rule management established |
| Documentation | Minimal to none | Basic documentation implemented |
| Version Control | Absent | Introduced |
| Threat Landscape Review | Little to no regular review | Regular reviews initiated |
| Personnel Training | Limited or none | Initial training provided |
|  |  |  |

**GOOD TO HAVES - NOT ESSENTIAL IN Tier 0, ESSENTIAL IN Tier 1**

**4. Prioritizing Detection Engineering Efforts with Limited Resources**

Detections follow this flow → Baselining queries → followed by → Proactive risk based/need based investigations - some ways to proceed in prioritizing are below:

* **Risk-Based Approach:** Focus detection engineering efforts on addressing the organization's most significant cyber risks 11. Identify the threats that pose the greatest potential impact to the business, considering factors such as data sensitivity, business continuity, and regulatory compliance. Prioritize the development of detections that can identify and alert on these high-risk threats.
* **Leveraging Threat Intelligence:** Actively utilize available threat intelligence resources to stay informed about the latest threats, attacker tactics, and emerging vulnerabilities 8. This intelligence can come from various sources, including vendor security blogs, industry reports, open-source threat intelligence feeds, and information shared within the security community. Focus detection efforts on the threats and techniques that are most relevant and actively being exploited in the current threat landscape.
* **Focusing on High-Value Assets:** Prioritize the development of detection rules that specifically monitor and protect the organization's most critical assets and data 10. These high-value targets might include sensitive customer data, intellectual property, critical infrastructure components, or executive accounts. Ensuring robust detection capabilities around these assets will significantly reduce the potential impact of a successful cyberattack.
* **Starting with Known Threats:** Begin by building detections for known threats and common attack techniques that are relevant to the organization's environment [User Query]. This provides a more concrete and manageable starting point than attempting to immediately detect novel or highly sophisticated attacks. Focus on implementing detections for well-documented tactics, techniques, and procedures (TTPs) that are known to be used by threat actors targeting organizations in their industry or with similar profiles.
* **Iterative Expansion:** Adopt an iterative approach to detection engineering 1. Start with basic, high-fidelity detections that address specific threats, and then iteratively expand and refine these detections over time as the team's knowledge of KQL, MDE, and the organization's environment grows. This allows for a more pragmatic and sustainable approach to building detection capabilities, acknowledging the current limitations of a small team with potentially limited initial expertise.

**5. Structuring and Organizing Your KQL Queries for Efficiency**

As the team develops more KQL queries, establishing a clear structure and organization for managing them will become increasingly important for maintaining efficiency and avoiding confusion. Implementing the following strategies can help:

* **Naming Conventions:** Adopt a consistent and informative naming convention for all KQL queries. This convention should ideally include key information such as the threat or technique being detected (consider using MITRE ATT&CK T-codes if the team is familiar with the framework), the primary data sources utilized by the query, and a version number to track revisions. For example, a query detecting phishing emails with malicious attachments using the EmailAttachmentInfo table might be named T1566\_Phishing\_MaliciousAttachment\_EmailAttachmentInfo\_v1. This allows for quick identification and understanding of the query's purpose.
* **Simple Repository:** Establish a centralized and easily accessible repository for storing all KQL queries. For a small team, this could be as simple as a dedicated folder within a shared document library like SharePoint or OneDrive. Alternatively, a shared OneNote notebook with a section for detection queries or a basic spreadsheet with columns for the query name and the KQL query syntax can also serve as effective initial repositories. The key is to ensure that all team members know where to find the latest versions of all queries.
* **Basic Documentation Template:** Create and utilize a standardized template for documenting each KQL query. This template should include essential fields such as the query name (following the established naming convention), a detailed description of the query's purpose, the name of the author, the date of creation, the date of the last modification, the complete KQL query syntax, the specific MDE data sources used, the initial hypothesis that the query was designed to test (if applicable), any analysis notes (particularly regarding observed false positives or tuning efforts), and a brief version history outlining any significant changes made to the query over time.
* **Categorization/Tagging:** If the chosen repository supports it, implement a system for categorizing or tagging queries. This could involve using folders within the document library, tags within OneNote, or dedicated columns in a spreadsheet. Consider categorizing queries based on the MITRE ATT&CK framework tactics and techniques, the specific type of threat being detected (e.g., Phishing, Malware, Credential Access), or the primary MDE data sources that the query relies upon (e.g., EmailEvents, DeviceProcessEvents). This will significantly improve the ability to search for and manage the growing collection of queries.
* **Version Control:** Even with a simple repository, it is crucial to implement basic version control practices 1. This could involve appending an incrementing version number to the query name each time a significant change is made (e.g., from \_v1 to \_v2). Additionally, maintaining a brief log of the changes made in the documentation for each version will provide valuable context. As the team's technical skills and the complexity of their detection rules increase, they may consider adopting a more robust version control system like Git, which offers features such as branching and merging for collaborative development.

Sources :