OWASP Vulnerability Management Guide (OVMG)

June 1, 2020

Table of Content

I. Foreword

The objective of this document is to bridge the gaps in information security by breaking down complex problems into more manageable repeatable functions: detection, reporting, and remediation. The guide solely focuses on building repeatable processes in cycles. When implementing, it is recommended to start “small” and then incrementally and continuously refine each task and sub-task in the Cycle. While you, as an individual or an organization, may not know all answers to the questions outlined in the OWASP Vulnerability Management Guide (OVMG or the guide), it should not prohibit your business from becoming more resilient through vulnerability management program adoption.

About OVMG

The document is organized as follows: There are three cycles (tricycle), each of which has a numeric value and color code. The tasks inside of each Cycle have the corresponding colors and numbers.

1 Detection #FB027F	2 Reporting #FDCC65	3 Remediation #66CCFE

Each Cycle is a domain that comprises four main processes. Each process is essentially a Task that includes a to-do list. The order of these lists is logical but could be adjusted to fit your objectives.

All tasks have “Inputs” and “Outputs.” For example, the task “Scope” feeds into multiple processes: set-up of the security tools for vulnerability testing, grouping the assets for scans and reports, prioritizing remediation, applying metrics in vulnerability reports, and defining what is acceptable and what is not. The “Outputs” of the Scope may be impacted by changes coming from the “Inputs.” This is imperative to remember! Your Scope changes as you receive feedback from reports and exceptions.

The cyclical nature of vulnerability management implies continuous process improvement, and it is crucial to understand how a single process feeds into other processes and how all tasks are interconnected across three domains. The official web page of the OVMG contains a GIF animation that illustrates connections among all tasks in the tricycle.

II. Guide

1 Detection Cycle

During the detection cycle, we conduct the tasks that support vulnerability tests in essential ways by defining the: who, what, where, why, and how. The principal activities are focused on defining and refining the scope after each round of the tricycle, getting tools ready and verifying their integrity, conducting tests, and verifying results.

1.1 Scope

1.1 TASK	INPUT	OUTPUT
Define/Refine scope	2.4 Reports 3.4 Exceptions	1.2 Tools 2.1 Assets Groups 2.2 Metrics 2.4 Reports 3.1 Prioritize 3.4 Exceptions

#	TO-DO	WHY
1.1.1	Know the enterprise risks	Whether your organization does or doesn't have a risk registry, you have to understand what risks worry your management the most and where those risks are coming from. Understand the magnitude of monetary losses, understand what may jeopardize the business your organization is in. Understand what may become grounds for potential exceptions.
1.1.2	Know operational constraints	Understand what may jeopardize your business due to inadequate procedures, processes, system failures, human errors, lack of talent, fraudulent or criminal activities. What are the legal, regulatory, and contractual requirements that your organization must meet? Gather information about the relevant policy. Do you need to create a vulnerability management policy or update it?
1.1.3	Know technical constraints	Know and understand the limits of your assets and interdependencies with regards to obsolete technologies. For example, some SCADA hardware may not work unless the OS supporting it is Windows XP.
1.1.4	Distinguish primary assets vs. secondary	Know the essential assets, the loss of which would be detrimental for business, as well as the supportive, secondary assets. For example, a production server for the customers and a financial server with the payroll data. Know the assets that are exposed to the public Internet, consider these assets as critical.

		When rolling out an enterprise-wide vulnerability management program, start with the critical assets, and then incrementally expand to all essential, or secondary assets, and all other assets.
1.1.5	Embed vulnerability management processes into enterprise processes	Promote incremental change to fight any incumbent inertia (or a push back) at your organization. Sometimes it's faster to build a new program on top of existing processes and refining the processes as you go. For example, by knowing the dates of the monthly patching window, you can aid your engineering team by providing vulnerability analysis before patching and after.
1.1.6	Build managerial support	You must have a managerial buy-in because a vulnerability management program will require the attention of several departments and multiple stakeholders. Make sure your management understands its importance and supports the vulnerability management program. If not, please review 1.1.1 and do some additional reading on enterprise risk topics. No business leader wants to incur losses.

End Goal: your management should give you sign-off on a specific vulnerability test in writing. Ideally, you should have a vulnerability management policy ready, but that might happen after you complete several rounds of OVMG. By completing the Scope task, you should be able to explain to your management and your peers why vulnerability testing is needed and how it benefits the business. You should be able to outline the next steps. You should understand the boundaries of vulnerability tests.

1.2 Tools

1.2 TASK	INPUT	OUTPUT
Optimize Tools	1.1 Scope 1.4 Confirm Findings	1.3 Run Tests 1.4 Confirm Findings

#	TO-DO	WHY
1.2.1	Determine the type of your test/scan	The scope defines targeted assets and determines what type of security test you'll conduct. The common choices are: • Network scans: credential vs. uncredentialed scans • Applications scans: static code analysis (SAST) vs. dynamic scans (DAST) • Business email security or Social Engineering (SE) security tests Network scans are suitable for detecting missing patches, misconfigurations, and default credentials on web servers and network devices. The credentialed scan usually provides more accurate results than non-credentialed. We tend to use non-credentialed scans for scanning assists exposed to the public Internet. Note, when you are rolling out the scans for the first time (and that may include a first time for some group of assets), check the "health" of assets before and after. While SAST analyzes the quality of code, DAST simulates real-world attacks. Note, DAST may cause some damage to the web application and underlying server. It would be wise to avoid running DAST in the production environment.

		Business email security tests, or phishing tests, are a way to engage the critical thinking of users and prevent click fatigue. SE tests are not very common but have been found to be a very effective way to raise self-awareness in employees. Note, retraining should be preceded by formal information security training.
1.2.2	Determine the frequency of your security tests	The scope should provide the input based on legal, regulatory, and contractual requirements that your organization must comply with. The most popular compliance framework for vulnerability management is PCI DSS.
1.2.3	Ensure the latest vulnerability feed	Subscribe to "patch Tuesday" emails from all your major vendors. Subscribe to the full disclosure database and other feeds where you can track all new CVEs. Ask the tool vendor how long it takes to update vulnerability definitions in their feed; it could be up to 1 or 2 weeks from the patch release.
1.2.4	Check if vulnerability exceptions exist	If you inherited the vulnerability scanner tool, make sure that some vulnerabilities are not exempt from showing up on the report.
1.2.5	Test your tool for integrity	You can scan your computer or other devices you are well familiar with and have access to. Cross-reference the output from your scanner with what is actually on the device. Does your scanner properly fingerprint your operating system or enumerate all URLs of a Web application? Were all applications running on your device enumerated? Alternatively, you can use the OWASP vulnerable applications to assess if you correctly set up your dynamic scanner for application tests. Check out the OWASP Juice shop or the OWASP Mutillidae.
1.2.6	Adjust your tools' settings, preferences, templates	Start safe and small, observe results, then increment and observe again. What is different? Does it add any value? Read help and feedback provided by the community around these security testing tools. Ensure that you are not inside your own bubble.
End Goal: you should be able to adjust your tools to fulfill the scoped objectives.		

1.3 Run Tests

1.3 TASK		INPUT	OUTPUT
Run Vulnerability Tests		1.2 Optimize Tools 3.2 Remediation	1.4 Confirm Findings 2.4 Reports
#	**TO-DO**	**WHY**	
1.3.1	Scan public IP addresses	Apply a non-credentialed scan, check for default passwords. The goal is to see what an attacker would see.	
1.3.2	Scan private subnets	Apply credentialed scans using service accounts. Using credential scans increases the rate of accuracy. Consider secure credential handling.	
1.3.3	Scan/test web applications	Find out how a web application could be exploited. Use a replica of the production for security testing.	

1.3.4	Scan/test mobile apps	Find out how users may exploit a production app.
1.3.5	Test users (phishing, social engineering training)	Users are the most valuable yet prone to Social Engineering assets. Use security testing to find out who is likely to click the malicious link or execute a malicious drop. Link the results to retrain users.

End Goal: you should be able to run vulnerability tests as planned.

1.4 Confirm Findings

1.4 TASK	INPUT	OUTPUT
Confirm Findings	1.2 Optimize Tools 1.3 Run Tests	2.4 Reports 1.2 Optimize Tools

#	TO-DO	WHY
1.4.1	Check if your test results have valuable data	The scan results could be incomplete, inconclusive, or contradictory. It may take some tweaking to find the right fit for each environment. Be sure to whitelist the IP associated with the scanner on the firewall side. Otherwise, the firewall might filter out any attempts to connect to various ports, meaning you will see all ports closed and no vulnerabilities. It is vital to ensure the integrity of your results before you share them with your management and teams.
1.4.2	Interpret and reconcile system/device fingerprinting across your tests	Take your time and go through the results, ensuring that device fingerprinting is representative of your environment and well defined. You might want to run the discovery scans before you start running vulnerability tests. Rerun the security tests as needed.
1.4.3	Determine that running services are what they are supposed to be	It is plausible that the tool may capture as a vulnerability software that is no longer in the system. You want to make sure that you adjust your tool settings to be a credible source of vulnerability discovery.
1.4.4	Find something that falls out of the pattern and investigate why	You'll be able to explain something out of ordinary if you spot it first and find a reasonable explanation based on facts (not your opinions though). Thus, you'll learn your tool better.
1.4.5	Randomly select vulnerabilities and confirm them with a different tool or manually	Every given vulnerability may have a level of certainty and risk. Some vulnerabilities are harder to replicate or prove, and some are harder to exploit. At the end of this exercise, you may improve your pen-tester skills and learn something new about a vulnerability that may help to give it a higher or lower priority and improve your reporting.

End Goal: understand the security test results; use the collected data to tune the vulnerability scanning tool for precision.

2 Reporting Cycle

The reporting cycle targets activities that help an organization understand vulnerability in a measurable way. The principal activities are focused on learning, categorizing and creating organizational, meaningful metrics that would become the cornerstone of vulnerability management reports. This should be followed by assigning work for remediation.

2.1 Asset Groups

2.1 TASK		INPUT	OUTPUT
Create Asset Groups		1.1 Scope 2.3 Audit Trail	1.3 Run Tests 2.4 Reports 3.1 Prioritize
#	**TO-DO**	**WHY**	
2.1.1	Determine functional asset groups	Understand what assets are mission-critical and what is not. Gather this information if the asset management tool is not available by talking to your management and coworkers. Think about how long the asset could be unavailable without causing damage to the business? Within a broader group of assets, web servers, for instance, create meaningful asset groups that could be used in vulnerability reports. Examples may include but not limited to location, department, and type of the asset (virtual vs. HD, cloud vs. data center, e.g.). Your guiding criteria should make sense to the audience you're reporting to.	
2.1.2	Determine asset groups by type of environment	Test your production, staging, and development environments, then compare vulnerability data of each environment. Do you see identical data or not? Differences may be indicative of governance issues. Grouping assets by the type of environment may be beneficial to prioritization.	
2.1.3	Determine asset groups by type of system	What OS bears the most of high severity vulnerabilities? Where are the problems concentrated? If an organization is a Windows shop, and the scan results indicate critical vulnerabilities on Apache servers, that would mean incompliance or lack of change management.	
2.1.4	Determine groups by CVE numbering authority or underlying technology	Understand what vulnerabilities are unacceptable for your organization to have. For example, group CVE-2017-0143, CVE-2017-0144, CVE-2017-0145, CVE-2017-0146, CVE-2017-0147, and CVE-2017-0148 into "EternalBlue/Petya" vulnerability group and track it.	
2.1.5	Determine groups by type of vulnerability	Apply OWASP Top 10 Web Application Security Risk. Network vulnerability types could be categorized but not limited: - Remote code execution - Weak cipher vulnerabilities - Obsolete/outdated software vulnerabilities - Information disclosure vulnerabilities - Privilege escalation - Default credentials	

		- Memory allocation/corruption

End Goal: you should know your environment enough to come up with the categories for your organizational assets.

2.2 Metrics

2.2 TASK	INPUT	OUTPUT
Define/Refine Metrics	1.1 Scope 2.4 Reports	2.4 Create Reports

#	TO-DO	WHY
2.2.1	Determine the amount and percentage of vulnerable assets	Distribute the determined amount and percentage across all functional groups. It would be significant to present in reports that some functional groups are more vulnerable than others.
2.2.2	Determine the amount and percentage of vulnerable assets by severity and CVSS	Repeat the step above and apply severity or CVSS score. Severity grading depends on the vulnerability scanner tool but usually corresponds with CVSS. In the end, you should have several graphs where axes Y are functional groups A, B, C, D, and axes X are severity ratings.
2.2.3	Determine the amount and percentage of new vulnerabilities:	During this step using the most recent and verified test results, you'll populate tables, graphs, charts for your report, applying the qualifiers from the left column (2.2.3.1 to 2.2.3.6.)
2.2.3.1	-by severity	Severity grading depends on a scanner's brand but corresponds with CVSS for network scanners. For web applications, it could be high, medium, and low.
2.2.3.2	-by functional groups	Apply your results from 2.1 and break down further or consolidate. For example, functional groups could be: - predefined by teams that support these assets: network, customer hosting, e.g.; - predefined by the type of the devices: webservers, ICS, workstations, IoT; - predefined by the location of the assets.
2.2.3.3	-by type of environment	For example: production, development, testing, enterprise network, public IPs, hosted applications.
2.2.3.4	-by type of system	Apply the results from 2.1. and aggregate data by an operating system.
2.2.3.5	-by CVE numbering authority	Apply the results from 2.1. and aggregate data by CVE number.
2.2.3.6	-by type of vulnerability	Please see 2.1.5.

2.2.4	Compare and analyze aging data by the severity of vulnerabilities and their share:	There are two aspects of vulnerability aging: the date of publishing a vulnerability (reflected in CVE number) and the time of discovery. Aging vulnerability data analysis will impact the priority for remediation work.
2.2.4.1	-enterprise-wide	Think about enterprise elements that have the oldest vulnerabilities.
2.2.4.2	-among all other vulnerable assets	An asset could be more susceptible to exploitation if aged vulnerabilities prevail. On the discovery side, if your report shows that the vulnerability was discovered 180 days ago, it can mean that the specific business process did not get fully adopted or lacks quality control.
2.2.4.3	-by functional groups	Aggregating this data by functional groups, namely which are responsible for remediation of asset groups, would be an accountability reference in your report.
2.2.4.4	-by type of environment	For example: a public infrastructure vs. private infrastructure, production vs. testing. Define what contrasts are essential to your organization.
2.2.4.5	-by type of system	For example, public servers, internal servers, network endpoints, workstations, IoT systems, SCADA systems. You can go more granular to distinguish DNS servers from email servers, e.g., or firewall interface from other network devices. You can identify vulnerability by the type of operating system.
2.2.4.6	-by CVE numbering authority	Look at your data and assess whether there is any disproportion among specific vulnerabilities.
2.2.4.7	-by type of vulnerability	Please see 2.1.5 and cross-reference the test results among functional groups. By doing that, you would be able to map certain types of risks that should be mitigated or accepted organizationally.
2.2.5	Draw out trends by count and percentage utilizing KPI that matter to your enterprise risks and compliance	As you repeat the OVMG tricycle, you'd be able to observe changes that put everything in perspective. Note, a downward trend may mean that we are doing a good job remediating vulnerabilities or that we don't detect them correctly. If that's a concern, consult with 1.4 Confirm findings.
2.2.6	Determine exploitability of vulnerable assets by severity; specify count, percentage, decrease or increase	Use a reputable source for exploitation and don't default to one source only, such as a KB reference. Some software companies may be less transparent than others.

End Goal: you should create (and later revise) metrics for vulnerability reports. These metrics should be consistent and make sense for the audience of the reports (management and teams assigned to do remediation work).

2.3 Audit Trail

2.3 TASK	INPUT	OUTPUT
Create Audit Trail	3.4 Exception 2.4 Reports 3.3 Investigate FP	2.1 Assets Groups 2.4 Reports

#	TO-DO	WHY
2.3.1	Use your organization's ticketing system	Remediation is essentially a work request. You should be able to comply with the existing work request process in use and track how long it takes to get the work done. Important - some organizations have automated patching processes; it doesn't mean that they are free from vulnerabilities. Thus, one could argue that the information security office acts as independent quality assurance by establishing a vulnerability management program.
2.3.2	Provide a summary of the issue	You want to be concise, to the point, and avoid adjectives other than relevant severity ratings: critical, high, severe, medium, moderate, or low.
2.3.3	Provide tool-based output	That would help to weed out the false positives or other errors.
2.3.4	Notify/assign the issue/ticket to the responsible teams or individuals	It is imperative to create a culture of accountability around remediation work. Assigning a person to a security issue may spark some political repercussions within an organization; you should be able to address the potential problems beforehand by communicating.
2.3.5	Make sure that your manager/CISO is aware	Therefore, it is critical to have your management backing your actions.

End Goal: create an audit trail for the remediation workload. Assign work or training to individuals who are responsible for vulnerability remediation (a code rewrite, a configuration fix, e.g.).

2.4 Reports

2.4 TASK	INPUT	OUTPUT
Create Reports	1.1 Scope 1.3 Run Tests 1.4 Confirm Findings 2.1 Assets Groups 2.2 Metrics 2.3 Audit Trail 3.2 Remediation 3.3 Investigate FP	1.1 Scope 3.1 Prioritize 3.2 Remediation 3.3 Investigate FP 3.4 Exception

#	TO-DO	WHY
2.4.1	Maintain a consistent frequency of reporting and use it to track changes	Consistent equals reliable. This step is vital to becoming a transparent and dependable source of information for the audience of vulnerability reports.
2.4.2	Aggregate and process collected data	Pull the original data without altering it and apply metrics that are useful to the audience. Be sure to document your process along the way to avoid accidental errors. Cross-reference data in a way that you could compare the results to make sure that they are correct.

2.4.3	Using CVSS, apply unique environmental traits to your vulnerability analysis	Although the CVSS score would be your common denominator, it is plausible that a lower risk score may be higher in your environment due to the exposure factor or aging vulnerability traits.
2.4.4	State vulnerability trends	What is different from the last month, week, quarter, year: is it better, is it worse, no change?
2.4.5	Hypothesize about these trends in one sentence	If we see a downward trend due to scanner failure, we want to state it in the report. If we see an upward trend because no remediation work has been done, it is the right place to communicate this.
2.4.6	In one paragraph, add your recommendations	Give practical advice on how to turn a high-risk environment into one of lesser risk by eliminating the “fill-in-the-blanks” vulnerabilities in the “fill-in-the-blanks” assets (subnets/systems/applications). Note: Delivery matters - we want to be as concise, pragmatic, and mission-oriented as possible.
2.4.7	Apply data sensitivity classification to your report	Consider what competitors or adversaries would pay for this information. Mark it as “confidential” at the minimum, reiterate a sensitivity mark on each page.
2.4.8	Make a shorter version (1-2 pages) of your report	This is where we sacrifice granularity to paint the broader picture: what do these vulnerabilities mean to the enterprise vulnerability spread and where are the problems concentrated. Make it more illustrative than verbose. Avoid using technical jargon or CVE numbers: “EternalBlue” could sound more familiar than CVE-2017-0143.
2.4.9	Submit both versions of the report to your manager/CISO	Use both electronic and verbal communication. You might want to store your reports on a shared encrypted drive and submit only a URL to the report.
2.4.10	Create and maintain your own vulnerability management repository for internal or external audit	Make sure to adhere to the auditability requirement. Make a secure storage location for the collected data and final reports. Be sure to document your process along the way to avoid accidental errors.
2.4.11	Be able to explain the details of vulnerability detection and the reporting process	Be transparent with your management and colleagues about data collection and data processing. Transparency plus consistency benefits your credibility.
End Goal: summarize security scanning results in a concise form that would be easy to understand. Share your reports with all who need to know. Keep vulnerability reports consistent in format and delivery.		

3 Remediation Cycle

The remediation cycle targets activities that should reduce or eliminate vulnerabilities or provide evidence as to why a particular vulnerability is believed not to be a threat.

Key activities are focused on defining priorities and terms of remediation work, discussing and documenting false-positives, and dealing with exceptions.

3.1 Prioritize

3.1 TASK		INPUT	OUTPUT
Prioritize Vulnerabilities		1.1 Scope 2.1 Asset Groups 2.4 Reports	3.2 Remediation
#	**TO-DO**	**WHY**	
3.1.1	Use your reports	To prioritize remediation work, you need to use metrics from your reports amplified by assets' criticality to your organization.	
3.1.2	Use trend analysis	What are the areas where the trend is going up and how do we normalize them? These areas should be prioritized.	
3.1.3	Use information from additional sources	It pays off to stay current of cybersecurity news: zero-days, significant ransomware exploits, etc. This news may shift the priorities of your remediation work.	
3.1.4	Apply other environmental factors	Your organization has daily, weekly, monthly, and quarterly priorities. Based on the function of each team, these priorities may be dominant or secondary. Think about how vulnerability management may feed into other teams' goals.	
3.1.5	Communicate to responsible and accountable stakeholders	In 2.3.1, we discussed the use of the ticketing system. Augment it with personal written and verbal communication. It largely depends on your organizational culture, but above all, human relations go a long way. You have to build support among your coworkers.	
End Goal: create a data-based argument for vulnerability prioritization.			

3.2 Remediation

3.2 TASK		INPUT	OUTPUT
Remediation Work		1.1 Scope 2.4 Reports 3.1 Prioritize	2.4 Reports 3.3 Investigate FP 3.4 Exception 1.3 Run Test
#	**TO-DO**	**WHY**	
3.2.1	Find the stakeholders responsible for remediation work	Determine if the assigned individual understands the issue and can actually resolve it.	
3.2.2	Communicate your findings via the tools and processes they use	If the assigned team is bound to internal procedures and knowledge sharing, obey to these rules.	

3.2.3	Establish a frequency and scope of patching, rewriting code, retraining people	Ideally, we'd like to align remediation frequency with our vulnerability testing frequency (1.3). You have to be ready for concessions at the early stages of implementing a vulnerability management program and push for improvement as you repeat these cycles monthly.
3.2.4	Establish a group of assets dedicated to remediation testing	For example, you've been told that a configuration fix was mass-applied -- test it right away and don't wait until another monthly cycle of detection.
3.2.5	Report your test results to the responsible stakeholders	For the audit trail, store the test results on a shared drive or append them to the ticketing system.
3.2.6	Use the ticketing system or change management system to resolve remediation management issues	It is plausible to be in a situation where planning is behind reality. Regardless of whether it's true or not, find a way to utilize the ticketing or change management system for your audit trail.
3.2.7	Always assign remediation work	No assignee or no deadline for necessary remediation work means that your audit trail is incomplete.
3.2.8	Include responsible, accountable stakeholders, and those who need to be informed on unresolved issues	Consult with your organizational RACI chart, or consider the following based on your knowledge: personnel who applies a remediation fix, personnel who approves a remediation work; personnel who needs to be aware whether remediation has been done or not; personnel who may be impacted by upcoming work and needs to be informed of it.
3.2.9	Use the frequency of your reporting cycle to follow up on open issues	You can update your reports with remediation statistics; you can count recurring vulnerabilities; you can show aging statistics in your asset groups.

End Goal: complete vulnerability remediation work. Note, remediation is not to be assumed until retested in detection (1.3 Run Tests). All vulnerability instances that couldn't be remediated should be identified and documented.

3.3 Investigate False Positives (FP)

3.3 TASK	INPUT	OUTPUT
Investigate False Positives	2.4 Reports 3.2 Remediation	1.2 Tools 2.4 Reports

#	TO-DO	WHY
3.3.1	Ensure the integrity of a claim	Obtain evidence from the source: a screenshot, a code output, etc.
3.3.2	Construct a repeatable business process	Based on what is acceptable within organizational culture, create a repeatable and fair process.

3.3.3	Document all FP submissions	Documenting false positives should be a part of the process. You can use your ticketing system; you can use your testing tool, or both, whichever would maintain an auditable trail. Be aware of the balance of transparency and confidentiality: include the parties on "a need to know basis."
3.3.4	Find a SMEs who can agree or argue a false positive claim	Find a third party who can confirm or disprove the claim. Find an SME outside of your organization and ask him/her to comment on the issue without revealing the sensitive details.
3.3.5	Set a time frame at which FP should be reevaluated	It could be six months or a year. Use your legal and compliance guidelines to establish the time frame.
3.3.6	Document each FP and store it in an auditable repository	You can store it on a shared drive, as long as it remains confidential, just be aware of your sharing settings. Could they be modified without you knowing it?
3.3.7	Create an appropriate policy	Once you have a consensus on the process with your immediate players, you should codify some principal points in your vulnerability management policy.
3.3.8	Communicate this policy to all employees	How to do this should be specified by your organizational governance.
End Goal: establish ground rules for how a vulnerability is evaluated as a false positive. Review evidence on a case-by-case basis. Periodically revisit and revise false-positive cases. The process should be transparent and not be abused.		

3.4 Exceptions

TASK 3.4	INPUT	OUTPUT
Control Vulnerability Exception Process	1.1 Scope 3.2 Remediation	1.1 Scope 2.3 Audit Trail

#	TO-DO	WHY
3.4.1	Find an executive authority to sign off on a cybersecurity exception	Vulnerability exceptions imply that particular vulnerabilities may not be fixed for some time. There should be some business justification for that. Hence, we need to start by defining who has the final authority to approve a vulnerability exception. In many cases, that would be a CISO; in some cases, it might be a CEO. It depends on your jurisdictions and applicable law statutes.
3.4.2	Establish ground rules for vulnerability exceptions	There should be a strong business justification. For example, heavy SCADA machinery, expensive or impossible to replace, which functions while being ten plus years out of manufacturing support.
3.4.3	Establish periodic reviews of vulnerability exceptions	We should rely on law and compliance to establish these periodic reviews. If you are ISO 27001 complaint entity, this would be six months.
3.4.4	Establish acceptable compensating controls	Controls that we should set up to prevent the vulnerability from an exploit. The compensating controls should be periodically reviewed. The frequency of reviews can come from compliance and legislation.

3.4.5	Document each exception and store it in the company's audit system	The ticketing system may be used for this as well, be aware of information sensitivity and apply labels accordingly.
3.4.6	Create an appropriate policy	You can add vulnerability exceptions to your vulnerability management policy, or you can create a new one.
3.4.7	Communicate this policy to all employees	How to do this should be specified by your organizational governance.
3.4.8	Have vulnerability exception solicitors asking the executive authority for an approval every time	If the vulnerability exceptions process is too easy – it could become a loophole. Whoever seeks an exception should solicit a higher authority to approve it.

End Goal: you must ensure that all non-compliance is approved by senior management and documented in the company-wide repository. Vulnerability exceptions must have an expiration date, after which they should be revised. Vulnerability exceptions must include compensating controls that prevent vulnerability exploitation.

III. Figures

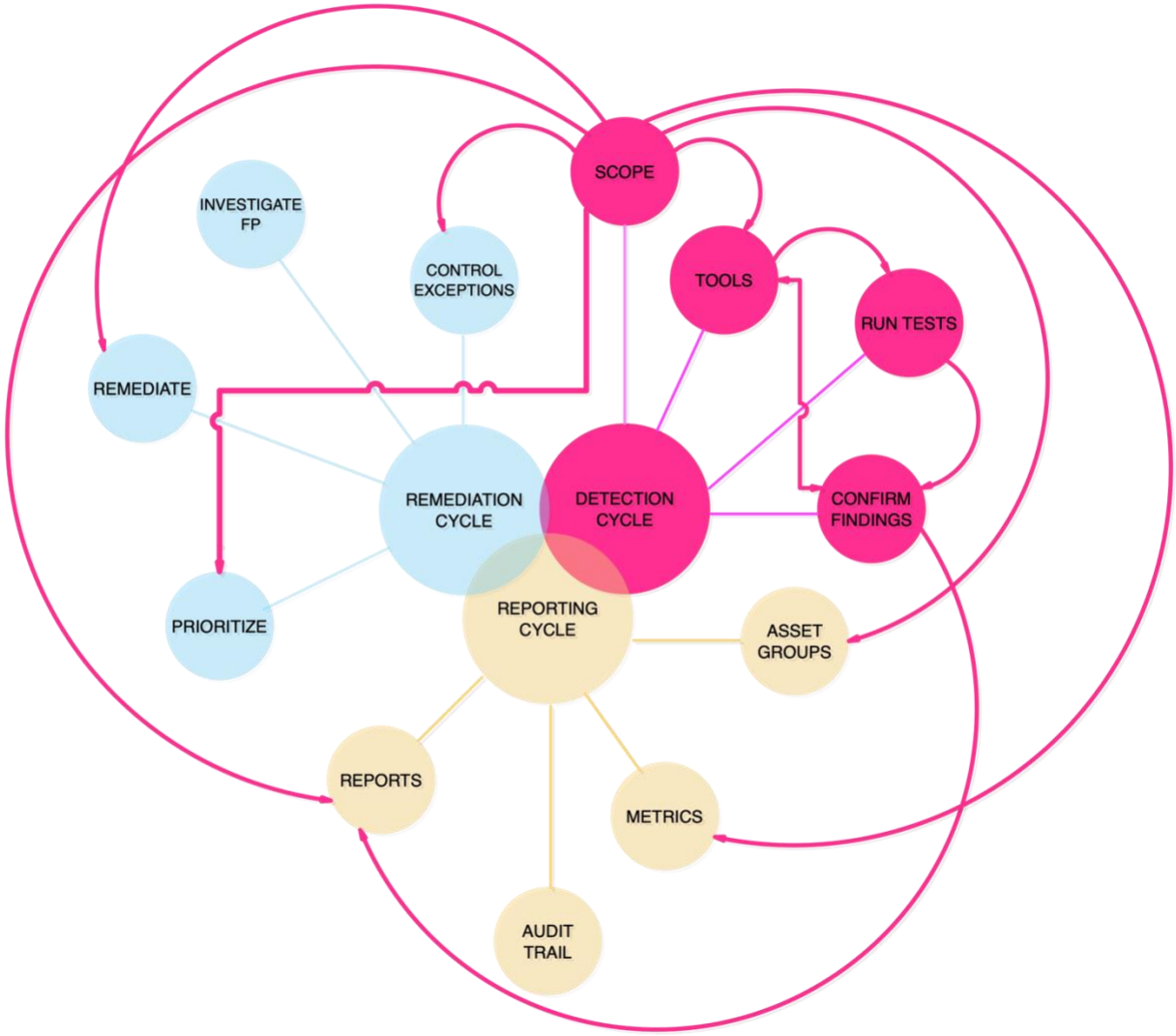

FIGURE A: DETECTION CYCLE INPUTS

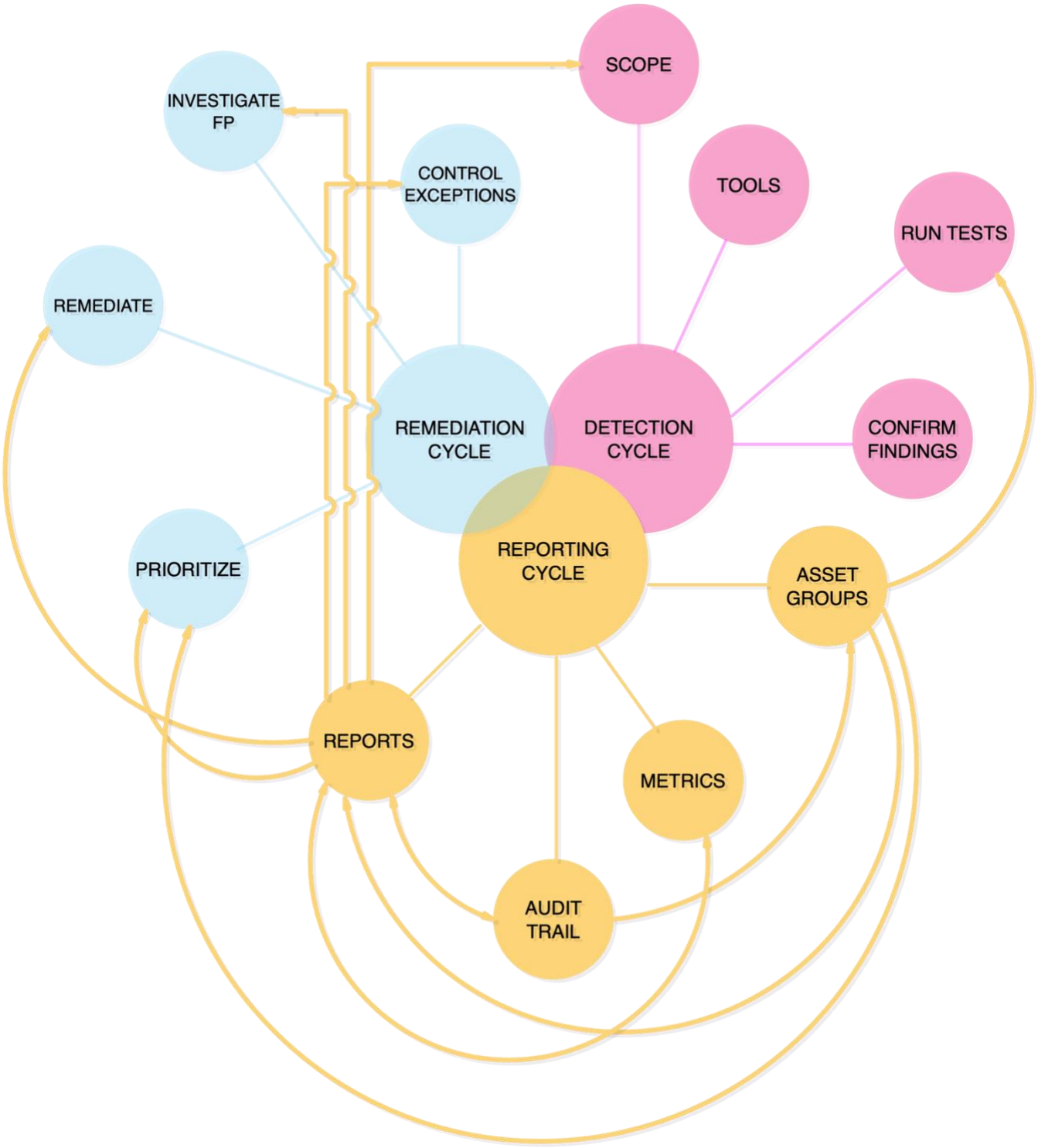

FIGURE B: REPORTING CYCLE INPUTS

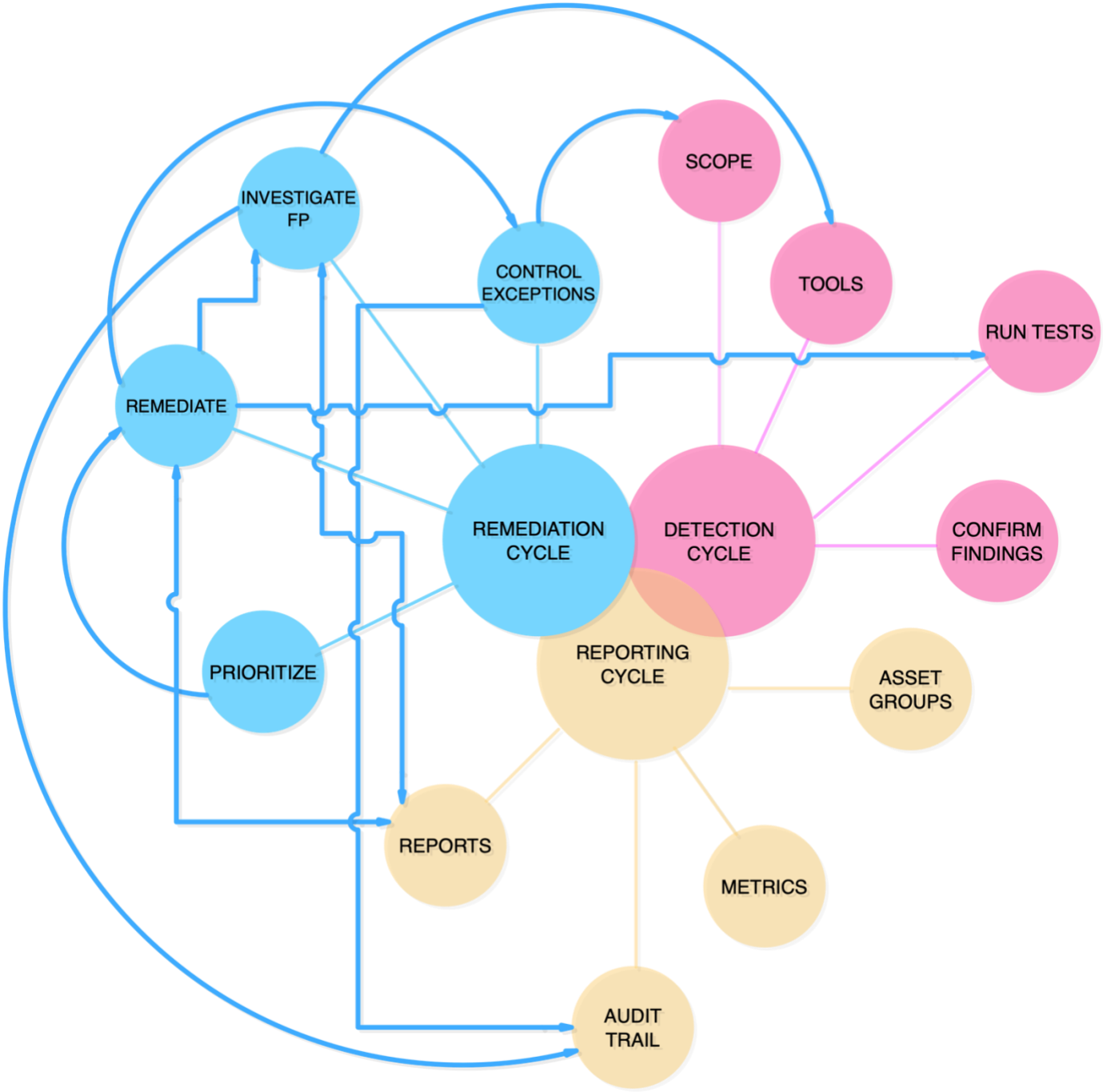

FIGURE C: REMEDIATION CYCLE INPUT

IV. Reference Table

Term	Definition
Asset	A device, a system, web or mobile application, a person
Audit(able) trail	Logs or records that provide chronological documentary evidence
CVE	Common Vulnerabilities and Exposures
CVSS	The Common Vulnerability Scoring System
FP	False Positive
KPI	Key Performance Indicator
KB	Knowledge Base
OWASP Top 10	https://owasp.org/www-project-top-ten/
OWASP Juice Shop	https://owasp.org/www-project-juice-shop/
OWASP Mutillidae	https://github.com/webpwnized/mutillidae
PCI DSS	Payment Card Industry Data Security Standard
RACI	Responsible, accountable, consulted and informed
SCADA	Supervisory control and data acquisition
SE	Social Engineering
SME	Subject Matter Expert