OWASP Vulnerability Management Guide (OVMG)

June 1, 2020

Table of Content

I. Foreword

The objective of this document is bridging the gap in information security by breaking down complex problems into more manageable repeatable parts: detection, reporting, and remediation. The guide is solely focused around 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), it shouldn’t disallow your business from becoming more resilient through vulnerability management program adoption.

About OVMG

The document is organized as follows. There are three cycles, each of which has a numeric value and color code:

1 Detection FB027F	2 Reporting FDCC65	3 Remediation 66CCFE

All processes inside of each cycle have a corresponding color and a number -- either 1,2, or 3, -- which doesn’t correspond with the numbering of the guide document.

Each Cycle is a domain that contains four main processes. Each process is essentially a Task that contains a to-do list. The order of the to-do list for the most part doesn’t matter; you can shuffle the sequence of the to-do list as you see a fit.

“Inputs” and “Outputs” are feeds for the continuous improvement of each task. For instance, 1.1 Scope influences multiple processes: how we set up our tools for vulnerability testing, how we group our assets for scans and reporting, how we prioritize remediation, what metrics we would use in our reports, what exceptions from remediation are acceptable and what are not. All of these Outputs of the Scope may change under influence of the Inputs of the Scope. It is very important to remember! Your Scope changes as you receive the feedback from reports and exceptions.

The cyclical nature of vulnerability management implies continuous process improvement and it is important to understand how a single process feeds into other processes and how all tasks are interconnected. The OWASP Vulnerability Management Guide official web page should contain a

simplified figure of what we call the OVMG tricycle: a cycle of detection, a cycle of reporting, and a cycle of remediation.

II. Guide

1 Detection Cycle

The detection cycle targets activities that establish vulnerability tests in the most essential ways: who, what, where, why, and how. The key 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 your organization must meet? Gather information about 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 assets that are absolutely essential, something there will be no business without, and 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 the enterprise processes	Promote incremental change to fight any incumbent inertia (or a push back) at your organization. Sometimes it is faster building a new program on top of existing processes and to refine the processes as you go. For example, by knowing the dates of the monthly patching window you can augment your engineering team by providing the 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 attention of several departments and multiple stakeholders. Make sure your management understands the importance and supports the vulnerability management program. If not, please review 1.1.1 and do some additional reading on enterprise risks topics. No business leader wants to incur losses.

End Goal: at the end of this activity your management should give you sign-off on a specific vulnerability test exercise in writing. Ideally, you have to have a policy ready, but that might not happen until you complete several rounds of tricycles. 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 for your teams and management. You also 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 targeted assets should be defined in the scope which largely influences what type of a test you want to conduct. The common choices are: Network scans: credential vs uncredentialed scans Applications scans: static code analysis (SAST) vs. dynamic scans (DAST) Network scans are good 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 that are exposed to the public Internet. While SAST analyzes the quality of code, DAST simulates real-world attacks. You may cause a damage to the web application and underlying server while	

		executing this type of a test. It would be wise to avoid running DAST in production environment. Note: When you are rolling out the scans for the first time (and that may include a first time for some group of assets) always check their "health" before and after.
1.2.2	Determine the frequency of your tests/scans	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 you all 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 properly set up your dynamic scanner for application tests. Check out OWASP Juice shop or 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 of your own bubble.
At the end of this activity you should be able to adjust the tools in use to fulfill the objectives of the vulnerability tests.		

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, test for default passwords, you would see what the attacker would likely see. This information is important.	
1.3.2	Scan private subnets	Apply credentialed scan using service accounts. Consider secure credential handling. This is important because you would see how your internal infrastructure could be exploited.	

1.3.3	Scan/test web applications	Use a replica of a production environment. You would find out how a production or pre-production application could be exploited when released.
1.3.4	Scan/test mobile apps	You would find out how a production app can be exploited by users who are sometimes hackers, researchers, or attackers.
1.3.5	Test users (phishing, social engineering training)	Although it's a special subject, users are the most expensive assets and could be tested if they are prone to SE or not. You would find out who may become a victim to SE.
At the end of this activity you should be able to run vulnerability tests in accordance with the scope.		

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 settings 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 important 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 to your environment and well defined. You might want to run the discovery scans before you start running vulnerability tests. Re-run the 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 on the system. You want to make sure that you adjust your tool settings to be the credible source of vulnerability discovery.
1.4.4	Find something that falls out of the pattern and investigate why	You would learn your tool better and you would 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).
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.

At the end of this activity you should be able to understand the security test results at use the collected information to tune the vulnerability scanning tool for improved precision or have a credible evidence that your tool doesn't need some fine tuning.

2 Reporting Cycle

The reporting cycle targets activities that help an organization to understand a vulnerability in a measurable way. The key activities are focused on learning assets that are subject to the security scans, creating a meaningful KPI that would become a cornerstone of vulnerability management reports, and assigning work for remediation.

2.1 Assets Groups

2.1 TASK		INPUT	OUTPUT
Create Assets 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	It is imperative to understand what assets are considered to be mission critical and what are not. You can gather this information if not otherwise available by talking to your management and coworkers. Think, how long the asset could be unavailable without causing damage to the business? Within a broader group, for instance, customers' servers, how would you break down your assets further and ultimately your scan results: by location, by department that maintain these servers, etc.? The answer lies in what makes more sense to the audience of your report: management and the people who will be fixing vulnerabilities.	
2.1.2	Determine asset groups by type of environment	Where are the problematic areas coming form: production, testing, development? If we are talking about web applications and we find more vulnerabilities in pre-production than in testing or development, it may be indicative of a process issues. Grouping assets by the type of environment may be beneficial to prioritization: what needs to be fixed ASAP because it may become a contractual liability.	
2.1.3	Determine asset groups by type of a system	The same reasoning as above: where are the problems coming from? If your organization is a Windows shop and for some reason the scan results indicate critical vulnerabilities on Apache servers, would that mean incompliance to the internal policy or lack of change management.	
2.1.4	Determine groups by CVE numbering authority or underlying technology	Consult with your compliance department, consult with your management. You would not get a precise CVE number, but you must understand what vulnerabilities is absolutely unacceptable to your organization to have. For	

		example, you may want to 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	Here is a good place to utilize your OWASP Top 10 knowledge. In a broader than web application vulnerabilities instance, it could be, but not limited: - Remote code execution - Weak cypher vulnerabilities - Obsolete/outdated software vulnerabilities - Information disclosure vulnerabilities - Privilege escalation - Default credentials - Memory allocation/corruption

At the end of this activity you should know your environment enough to come up with broad categories for your organizational assets, which may include IP ranges, applications, departments, or teams.

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 amount and percentage of vulnerable assets	Distribute this amount and percentage across functional groups. It would be significant to present in reports that some functional groups are more vulnerable than others.
2.2.2	Determine 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. At the end, you should have several graphs where axes Y are functional groups A, B, C, D and axes X are high severity (CVSS 10-7.0), medium severity (CVSS 6.9-4.0), or low severity (3.9-0) vulnerabilities.
2.2.3	Determine amount and percentage of new vulnerabilities:	During this step, with the most recent and verified tests data, you populate tables, graphs, charts (depends on a desirable outcome) 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, low.
2.2.3.2	-by functional groups	Apply your results from 2.1. For example, functional groups could be network devices, databases, webservers, ICS, workstations, IoT devices. Or, functional groups could be broken down by the location of the assets: Delhi, Kolkata, Bangalore, Mumbai.

2.2.3.3	-by type of environment	Apply your results from 2.1. For example: production, development, testing, enterprise network, public IPs, hosted applications.
2.2.3.4	-by type of system	Apply your results from 2.1. e.g., an operating system.
2.2.3.5	-by CVE numbering authority	Apply your results from 2.1. Aggregate by CVEs that are more often repeated across your test results.
2.2.3.6	-by type of vulnerability	Please see 2.1.5.
2.2.4	Compare and analyze aging data by severity of vulnerabilities and their share:	There are two aspects of vulnerability aging: the date of publishing (relates to CVE numbering) and the date of discovery. Aging vulnerability data should impact how soon vulnerability remediation will be prioritized.
2.2.4.1	-enterprise wide	Ask yourself, what enterprise elements may have the oldest vulnerability and why.
2.2.4.2	-among all other vulnerable assets	The asset could be more susceptible to exploitation if it contains more aged vulnerabilities. On the discovery side, if your report shows that the vulnerability has been discovered 180 days ago, it can mean that the certain business process did not get fully adopted or lacks a quality control.
2.2.4.3	-by functional groups	Aggregating this data by the functional groups (who is responsible for which segment of remediation) would be an accountability reference in your report.
2.2.4.4	-by type of environment	For example, a public infrastructure vs. private infrastructure, Linux vs. Windows, production vs. testing. You need to define what contrast is important 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 also further distinguish the type of system vulnerability data by location.
2.2.4.6	-by CVE numbering authority	You should look at your data and be able to tell if there is any disproportion among certain kinds of vulnerability. Do all functional groups suffer from the same vulnerability or only some and what would be the underlying reason?
2.2.4.7	-by type of vulnerability	Please see 2.1.5 and cross reference this data among functional groups and enterprise wide. By doing that, you would be able to map certain types of risks that should be mitigated or accepted.
2.2.5	Draw out the trends by count and percentage utilizing KPI that matter to your enterprise risks and compliance	As you are running further rounds of the tricycle, you would be able to observe a change that should 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 anymore. 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	It is important to remember, that we should use a reputable source for exploitation, and we should not default to one source only, such as a KB reference, since one software company may be less transparent than others.

At the end of this activity you should come up with metrics you can use consistently in your reports that would make sense for your audience, management, and your organization.

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 information security office acts as an 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 tools'-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	This is imperative to create a culture of accountability around remediation. Assigning a person to a security issue may spark some political repercussions within an organization; you should be able to address potential issues 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.

An end-goal of this activity is to create an audit trail which would manifest in a workload for teams or individuals who are responsible for vulnerability remediation: a code re-write, patching, or training. This is a prerequisite to 3.2 Remediation work.

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	1.1 Scope 3.1 Prioritize 3.2 Remediation 3.3 Investigate FP

		2.2 Metrics 2.3 Audit Trail 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 the changes	Consistent equals reliable. It is very important to become a reliable and transparent 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 the 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 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 the 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, weak, 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 none of our tickets were worked on, it is a good place to state it.
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” 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	Think, what would competitors or adversaries 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	That’s where we sacrifice granularity to paint the broad picture: what is vulnerability spread enterprise wide and where are the problems are concentrated. Make it more illustrative than verbose. Avoid using technical jargon or CVE numbers: “EternalBlue” could 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 the 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 the 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 the vulnerability detection and reporting process	Be transparent with your management and colleagues about the data collection and data processing. Transparency plus consistency benefits your credibility.

An end-goal of this activity is to summarize security scanning results in a concise form that would be easy to read for non-technical managerial personnel. Share your reports with all who need to know. Make your reports consistent in format and delivery.

3 Remediation Cycle

The remediation cycle targets activities that should reduce or eliminate vulnerabilities or provide evidence why a particular vulnerability is believed to not be a threat.
The key activities are focused on defining what priorities are, what should be patched and how soon, where the security scanner produced inaccurate results, and how to deal 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 the metrics from your reports amplified by assets' criticality in your organization.	
3.1.2	Use your trend analysis	What are the areas where the trend is going up and how to normalize these areas? We would need to prioritize this work.	
3.1.3	Use information from additional sources	It pays off to stay abreast of cyber security news: zero days, important ransomware exploits, etc. This news may shift 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 how vulnerability management may feed in 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 people you are working with.	

At the end of this activity, have a clear understanding with underlying reasoning as to what should be remediated ASAP, what should be remediated in a week, in a month, and so on.

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	In 2.3.1 we created a work ticket. In 3.1.5 we have had a meeting with the engineering team or its leadership. But who is actually going to apply a fix? Does he/she know what to do?
3.2.2	Communicate your findings via the tools and processes they use	If the team who is assigned to do remediation work is bound to their own set of rules and procedures (a change management, a test for patching) respect these rules.
3.2.3	Establish a frequency and scope of patching, rewriting code, retraining people	Ideally, we'd like to align this frequency with our testing cycle (1.3) but it may be not possible due to resource constraints. You have to be ready for concessions at the early tricycles and push for improvement as you repeat these cycles on a monthly basis.
3.2.4	Establish a group of assets dedicated for remediation testing	If you've been told that a configuration fix was mass applied, you might want to test it right away and not wait until another monthly cycle of detection. That could be something you agree to in advance with your coders or engineers.
3.2.5	Report back your test results to the responsible stakeholders	For transparency and an audit trail, store it on the shared drive or append to the ticketing system.
3.2.6	Use the ticketing system or change management system to resolve the remediation management issues	It is plausible to be in a situation where the planning is behind the 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 these tickets	No assignee means your audit trail is incomplete.
3.2.8	Include responsible, accountable, stakeholders, and who needs to be informed on unresolved issues	Consult with your RACI chart, otherwise consider who is applying a remediation fix, who is approving a remediation work, who needs to know whether remediation has been complete or not.
3.2.9	Use the frequency of your reporting cycle to follow up on open issues	You can update your reports with remediation stats; you can count recurring vulnerabilities; you can show aging statistics among functional areas.

At the end of this activity you should have vulnerability remediation work done by the assigned teams or individuals. Remediation should not be assumed until confirm by 1.3 Run Tests. All vulnerability instances that cannot be remediated should be identified either as FP or exceptions 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 integrity of a claim	Obtain the evidence from the source: a screen shot, a code output, etc.	
3.3.2	Construct a repeatable business process	Based on what is acceptable within the organizational culture, create a repeatable process that also would be considered as fair by your counterparts.	
3.3.3	Document all FP submissions	That 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 an SMEs who can agree or argue a false positive claim	Find a third party who can confirm or disprove the claim. You may find this SME outside of your organization and ask him/her to comment on the issue without revealing much of the details.	
3.3.5	Set a time frame at which FP should be reevaluated	It could be 6 months or a year. Use your legal and/or 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 the 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 that should be specified by your organizational governance.	

Establish ground rules for how one can assert that a discovered vulnerability is NOT an actual vulnerability. Review the evidence that suggests that an instance is not a vulnerability. Periodically review this evidence. False Positives should not mean “off the hook”.

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 cyber security exception	Vulnerability exception implies that certain vulnerabilities may not be fixed for some period of time. There should be some business justification to that, hence we need to start by defining who has a 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 jurisdiction and applicable law statutes.
3.4.2	Establish ground rules for vulnerability exceptions	There should be a strong business justification. Examples, a heavy SCADA machinery that functions being 10 plus years out of support.
3.4.3	Establish periodic reviews of vulnerability exceptions	We should rely on law and compliance establishing these periodic reviews. If you are ISO 27001 complaint that would be six months.
3.4.4	Establish acceptable compensating controls	In other words, what should be done to prevent the vulnerability from exploit. The compensating controls should be periodically reviewed (every six months).
3.4.5	Document each exception and store it in the company's audit system	The ticketing system may be used for that 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 that 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 easy that may become a loophole. Whoever seeks an exception should solicit the higher authority to approve it.

At the end of this activity, you must ensure that all non-compliance is approved by senior management and documented in the company-wide repo. All vulnerability non-compliance must have an expiration date.

III. Figures

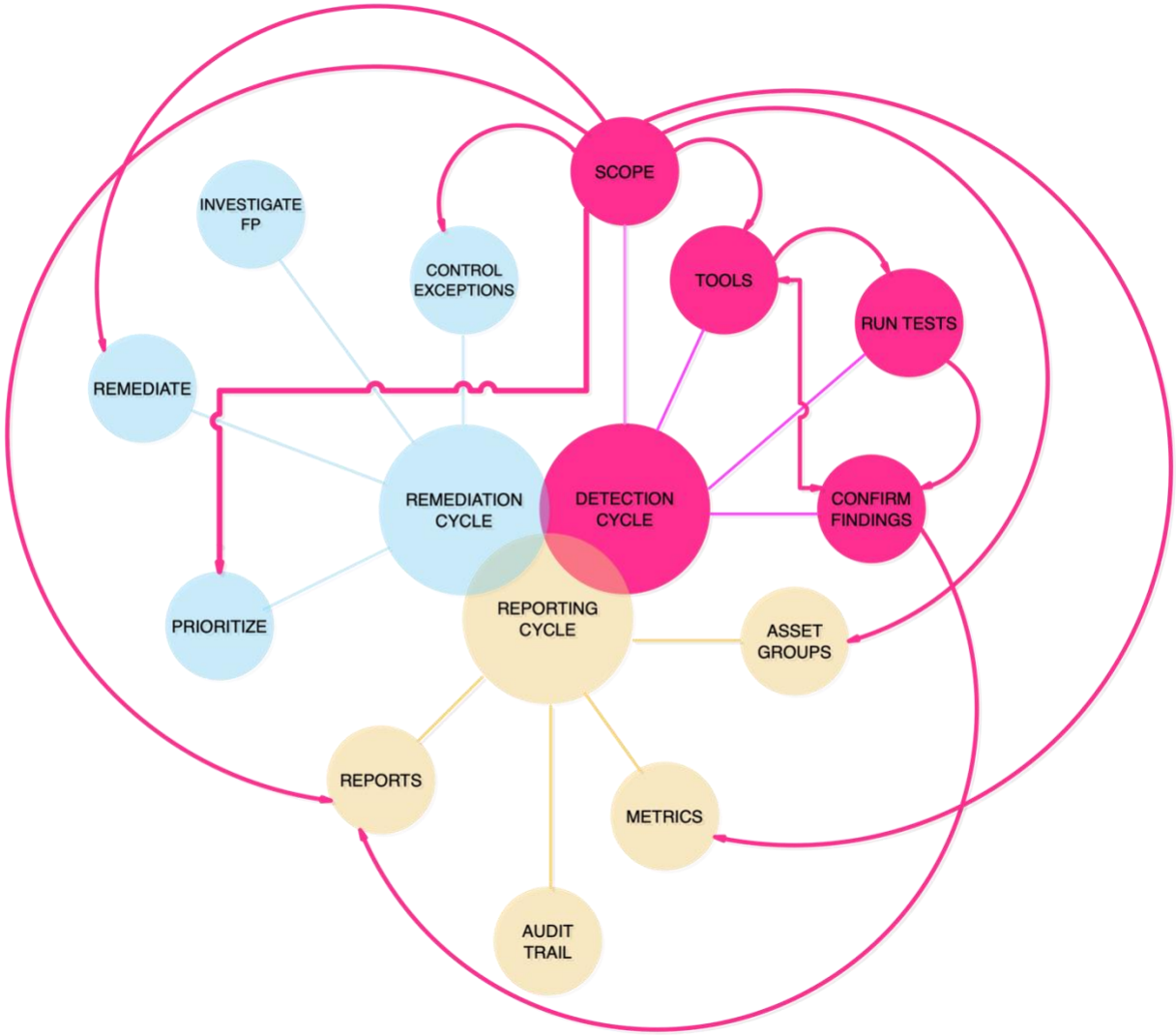

FIGURE A: DETECTION CYCLE INPUTS

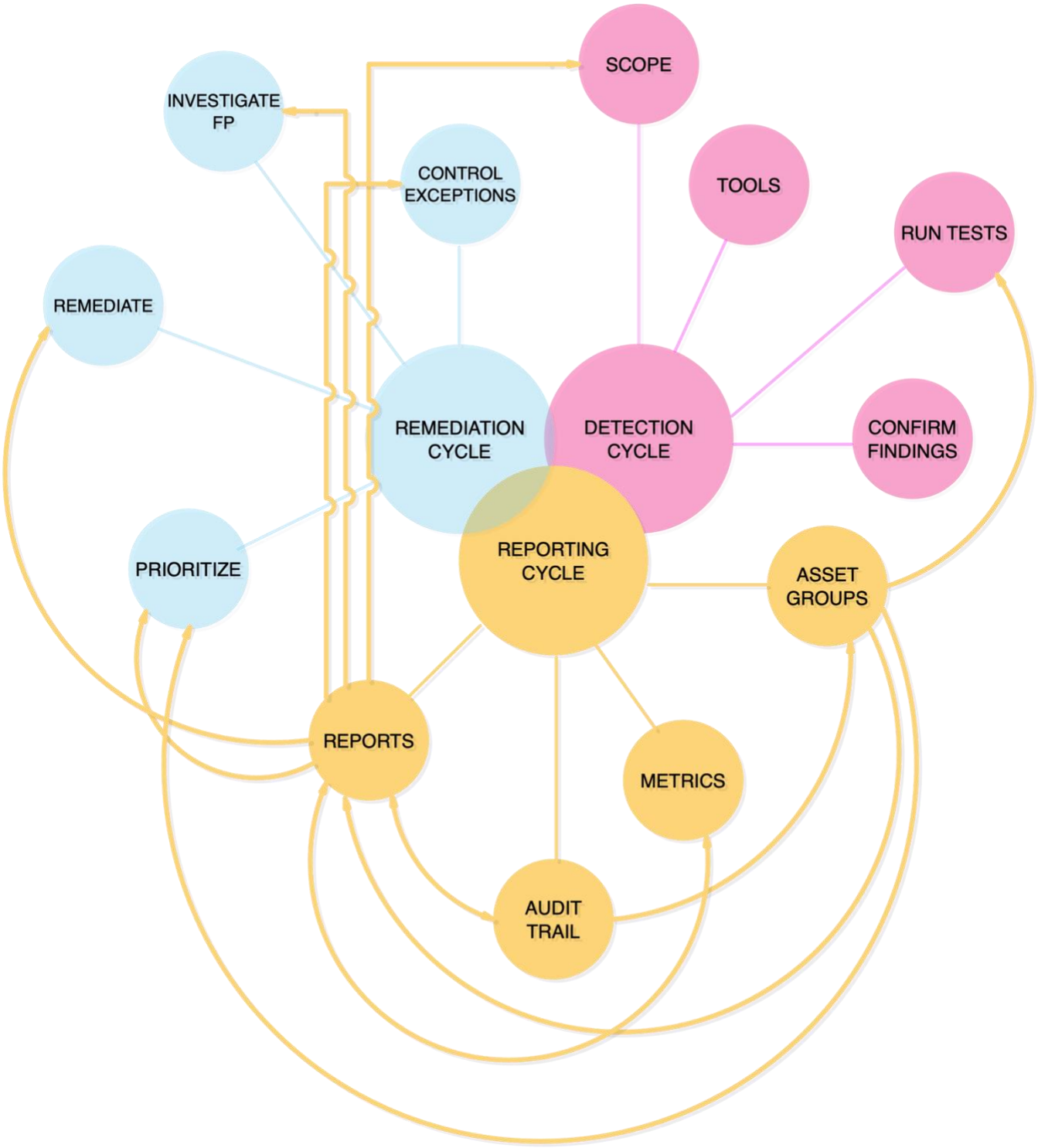

FIGURE B: REPORTING CYCLE INPUTS

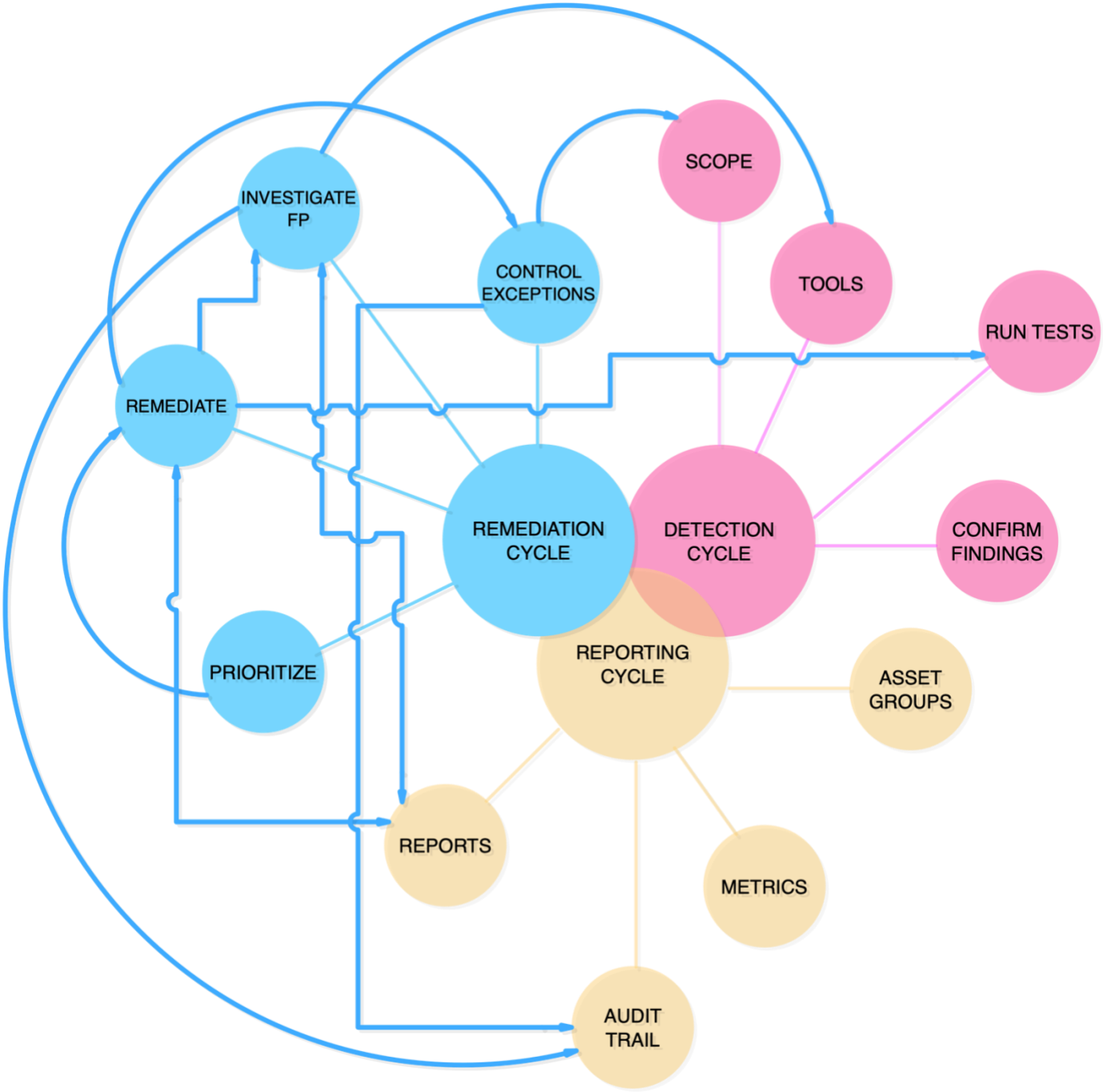

FIGURE C: REMEDIATION CYCLE INPUT

IV. Reference Table

Term	Definition
Asset	A device, a system, a 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
SME	Subject Matter Expert