Vulnerability Scanning and Compliance

Martin Stanek

2025

Table of Contents

Vulnerability Scanning

Configuration Audit / Compliance

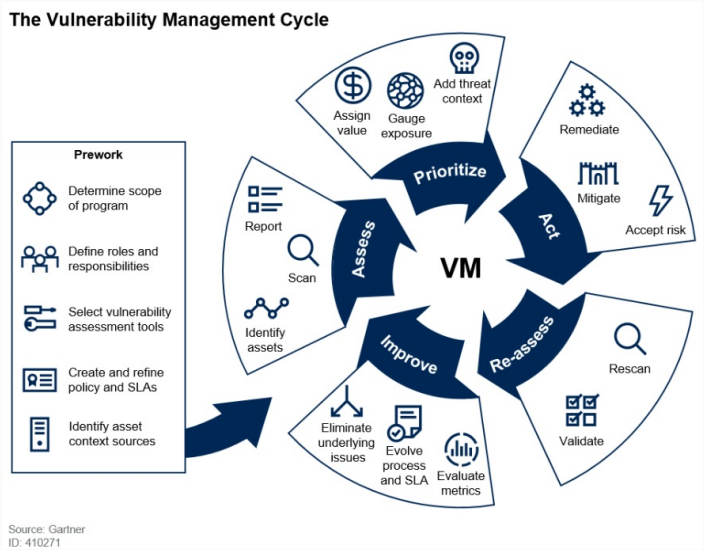
Benchmarks for Configuration Audits

Examples

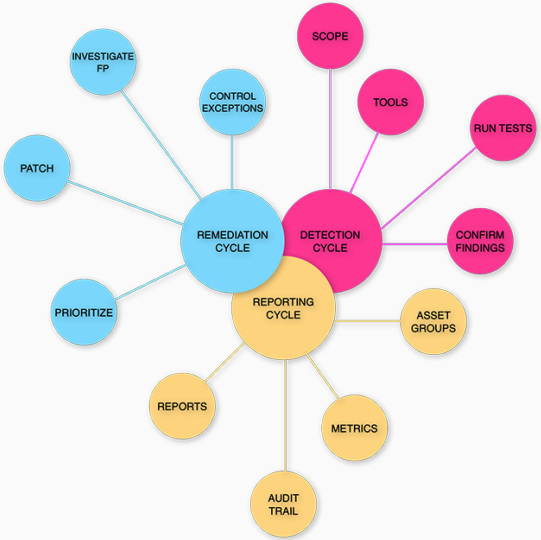
Introduction

- continuous stream of new vulnerabilities
- vulnerability management (VM) process (scanning is only a small part)
- NIST Cybersecurity Framework V2.0 (2024)
  - Risk Assessment (one of 22 categories)
    - Vulnerabilities in assets are identified, validated, and recorded
    - Potential impacts and likelihoods of threats exploiting vulnerabilities are identified and recorded
    - Threats, vulnerabilities, likelihoods, and impacts are used to understand inherent risk and inform risk response prioritization
    - Processes for receiving, analyzing, and responding to vulnerability disclosures are established

Vulnerability Management (according to Gartner)



Vulnerability Management (according to OWASP)



Maturity Model

- Capability Maturity Model
  - originally for SW development
  - applied to different IT processes – overall (such as cybersecurity) and partial (VM)
- usually 4-6 levels (CMMC – Cybersecurity Maturity Model Certification):

generic

1. Initial
2. Repeatable
3. Defined
4. Capable
5. Efficient

DoD CMMC 1.0

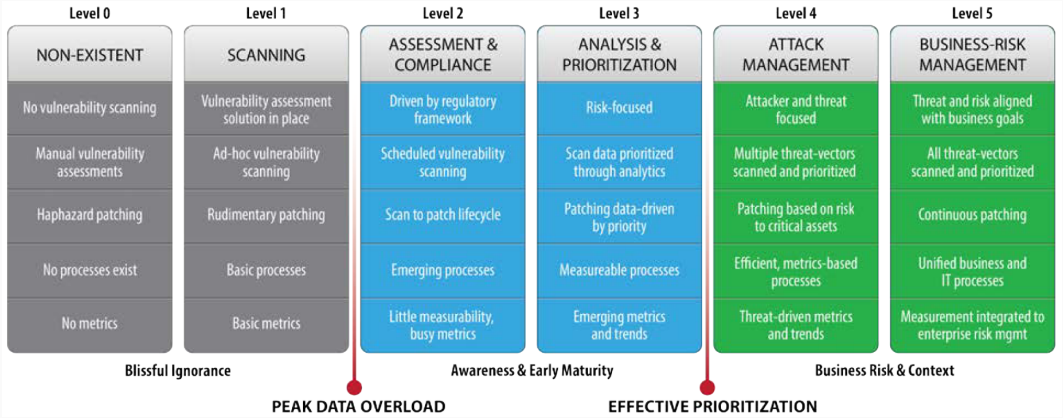
1. Performed (Basic Cyber Hygiene)
2. Documented (Intermediate C.H.)
3. Managed (Good Cyber Hygiene)
4. Reviewed (Proactive)
5. Optimizing (Advanced)

DoD CMMC 2.0

1. Level 1 (15)
2. Level 2 (110)
3. Level 3 (135)

requirements

Threat and Vulnerability Management Maturity Model (Core Security [1])



Sources – vulnerabilities

- NIST NVD (National Vulnerability Database)
- CERT/CSIRT Advisories and Alerts
  - US-CERT, CERT-EU, (SK-CERT, CSIRT.SK)
- SW vendors security advisories/patches, e.g.
  - Red Hat / Cisco / Oracle ... Security Advisories
  - Microsoft ... Security Update Guide / KB
  - Android Security Bulletin
- other repositories, e.g.
  - Vulners, VulDB, Exploit-db
  - Rapid7 Vulnerability & Exploit Database
  - IBM X-Force Exchange
  - Snyk Vulnerability Database (+ SW libraries)

Vulnerability Scanning – what and how

- web application vulnerabilities scanning
  - Burp Suite, OWASP ZAP, Invicti, etc.
- OS and 3rd party application scanning
  - Tenable, Qualys, Rapid7, OpenVAS, etc.
- consider
  - unauthenticated vs. authenticated scans
  - agent vs. agentless scanning
  - updates
  - air gapped systems

Priorities – what to fix first

- findings
  - large number of findings – What to fix first/immediately?
  - some (new) finding – How fast should we fix it?
- asset valuation/criticality, risk-based approach, "business aligned"
- vulnerability vs. exploit
- focus on exploitable vulnerabilities
  - Known Exploited Vulnerabilities Catalog (CISA)
- often a combined score based on
  - threat intelligence, exploit availability, vulnerability metadata (such as CVSS score), and asset criticality

Typical problems – incomplete coverage

- some systems are not scanned
  - inaccessible (e.g. due to firewalls)
  - unaware about their existence
  - specialized HW and appliances
- some applications are not identified
  - manually installed/compiled
  - non-standard locations
- some vulnerabilities are not included in the scanner's database
- custom applications (in-house or 3rd party developed)
- authenticated vs. non-authenticated scans

Other typical problems

- false positives
  - tests are just a SW (detection mistakes)
  - backported fixes
- remediation not available/impossible
  - it might break something
  - not fixed yet, fixed in *wrong* repository, *not completely* etc.
- patching is laborious, never-ending process
  - every month/quarterly/...
  - testing

Configuration Audit / Compliance

- Secure configuration
  - operating system, databases, web and application servers, etc.
  - define/adapt/choose a suitable security standard
  - check compliance with the standard and remediate deviations
- Compliance
  - liability (legal responsibility)
  - non-compliance can indicate a weakness

Secure Configuration – Sources

- CIS Benchmarks (Center for Internet Security)
- DISA (Defense Information Systems Agency)
  - STIGs – Security Technical Implementation Guides
- NIST – National Checklist Program Repository
  - defined in NIST SP 800-70 rev. 4
  - repository compiled from various sources
- Vendor specific guides
- Industry/legislation-specific standards and requirements
  - PCI DSS, HIPAA
  - might contain specific requirements for configuration

Problems with benchmarks/baselines

- missing benchmarks
  - recent OS/software versions
  - platforms and SW not considered by authorities
- differences in benchmarks – depth/coverage, recommended settings
- deviations – your environment is unique
  - decide, document, customize audit tools/templates, revise
- problems with automation
  - some requirements cannot be verified automatically
  - some benchmarks are not supported by tools

Automation

- vendor and product specific tools, e.g.
  - Oracle: Database Security Assessment Tool
  - Microsoft: Security Compliance Toolkit
- independent tools
  - vulnerability scanner/management (Qualys, Rapid7, Tenable)
  - SCAP Compliance Checker (SCC) from DISA
  - OpenSCAP, CIS-CAT, Lynis, etc.
- standardization effort – SCAP

Components for Automated Verification

- SCAP (Security Content Automation Protocol, NIST)
  - framework of specifications
- OVAL (Open Vulnerability and Assessment Language, MITRE → CIS)
- XCCDF (Extensible Configuration Checklist Description Format, NIST)
- CPE (Common Platform Enumeration, MITRE → NIST)
- CVE (Common Vulnerabilities and Exposures, MITRE)
- CVRF (Common Vulnerability Reporting Framework, ICASI → FIRSTs)
- Other:
  - CCE (Common Configuration Enumeration)
  - ARF (Asset Reporting Format), etc.

SCAP

multi-purpose framework of specifications that support automated configuration, vulnerability and patch checking, technical control compliance activities, and security measurement

- version 1.3 (NIST SP 800-126 Rev.3)
- plans for SCAP v2 (simplify authoring SCAP content, etc.)
  - no visible progress
- data streams collections:
  - checklists (XCCDF)
  - checks (OVAL)
  - dictionaries (CPE)
- supported by various security products (e.g. OpenSCAP, SCC)
- problems: complicated (XML mess), hard to tailor, incomplete coverage

XCCDF

specification language for writing security checklists, benchmarks, and related kinds of documents

- tree structure of a benchmark $\rightarrow$ profiles, values, groups
- group $\rightarrow$ values, groups, rules
- profile: named tailoring for a benchmark
- group: descriptive information about a portion of a benchmark
- value: named parameter that can be substituted into properties of other elements
  - account names, umask values, password parameters (e.g. length), etc.
- rule: defines a single item to be checked as part of a benchmark
- use STIG Viewer or open-scap.org for better reading

XCCDF example – benchmark, profiles

```
<xccdf:Benchmark id="xccdf_mil.disa.stig_benchmark_Oracle_Linux_8_STIG">
  <xccdf:title>Oracle Linux 8 STIG SCAP Benchmark</xccdf:title>

  <xccdf:Profile id="xccdf_mil.disa.stig_profile_MAC-1_Classified">
    <xccdf:title>I - Mission Critical Classified</xccdf:title>
    <xccdf:select idref="xccdf_mil.disa.stig_group_V-248519" selected="true" />
    <xccdf:select idref="xccdf_mil.disa.stig_group_V-248520" selected="true" />

  <xccdf:Profile id="xccdf_mil.disa.stig_profile_MAC-1_Public">
    <xccdf:title>I - Mission Critical Public</xccdf:title>
    <xccdf:select idref="xccdf_mil.disa.stig_group_V-248519" selected="true" />
    <xccdf:select idref="xccdf_mil.disa.stig_group_V-248520" selected="true" />
```

XCCDF example – group, rules

```
<xccdf:Group id="xccdf_mil.disa.stig_group_V-248554">
  <xccdf:title>SRG-OS-000206-GPOS-00084</xccdf:title>
  <xccdf:Rule id="xccdf_mil.disa.stig_rule_SV-248554r779228_rule" weight="10.0"
    severity="medium">
    <xccdf:title>The OL 8 "/var/log/messages" file must have mode 0640 or less
      permissive.</xccdf:title>

<xccdf:check>
  <xccdf:check-content-ref name="oval:mil.disa.stig.ol8:def:121"
    href="U_Oracle_Linux_8_V1R4_STIG_SCAP_1-2_Benchmark-oval.xml" />
```

- other info included: description, fixtext, reference, etc.

OVAL

language for assessment and reporting on the state of computer systems

- used for these steps in system assessment:
  - represent system information
  - express specific machine states
  - report the results of an assessment
- OVAL Repository and Registry of external OVAL repositories
  - in some cases narrowed focus (vulnerabilities, security advisories)
  - alternatives (Ansible etc.)
- structure
  - definition: what should be checked and what is expected, one or more tests
  - test: relationship between object and zero or more states
  - object: what should be collected
  - states: expected values from an object
  - variables: group one or more values for consistent reference

OVAL example – inventory (Microsoft Windows Server 2016 is installed)

```
<registry_test ... id="oval:org.cisecurity:tst:1883">
    <object object_ref="oval:org.mitre.oval:obj:5590" />
    <state state_ref="oval:org.cisecurity:ste:1519" />
</registry_test>
<registry_object ... id="oval:org.mitre.oval:obj:5590">
    <hive>HKEY_LOCAL_MACHINE</hive>
    <key>SOFTWARE\Microsoft\Windows NT\CurrentVersion</key>
    <name>ProductName</name>
</registry_object>
<registry_state ... id="oval:org.cisecurity:ste:1519">
    <value operation="pattern match">
  ^[a-zA-Z0-9\(\)\s-]*2016\s[a-zA-Z0-9\(\)\s]*$</value>
</registry_state>
```

OVAL example – compliance (Telnet Client feature is not installed)

- Microsoft Windows 10 STIG Benchmark – Ver 1, Rel 18 (DISA)

```
<file_test ... id="oval:mil.disa.fso.windows:tst:388900"
  check_existence="none_exist">
  <object object_ref="oval:mil.disa.fso.windows:obj:388900" /> </file_test>
<file_object ... id="oval:mil.disa.fso.windows:obj:388900">
  <path var_ref="oval:mil.disa.fso.windows:var:388700" .../>
  <filename datatype="string">telnet.exe</filename> </file_object>

<local_variable id="oval:mil.disa.fso.windows:var:388700" ... datatype="string">
  <concat>
    <object_component item_field="value"
       object_ref="oval:mil.disa.fso.windows:obj:388601" />
    <literal_component>\System32</literal_component>
  </concat>
</local_variable>
```

OVAL example – vulnerability (CVE-2020-0543, Intel CPU microcode problem)

- Debian security OVAL repository

```
<dpkginfo_test ... id="oval:org.debian.oval:tst:19407">
  <object object_ref="oval:org.debian.oval:obj:2240"/>
  <state state_ref="oval:org.debian.oval:ste:13293"/>
</dpkginfo_test>

<dpkginfo_object id="oval:org.debian.oval:obj:2240" >
  <name>intel-microcode</name>
</dpkginfo_object>
<dpkginfo_state id="oval:org.debian.oval:ste:13293" >
  <evr datatype="debian_evr_string" operation="less than">
  0:3.20200609.2~deb10u1</evr>
 </dpkginfo_state>
```

CPE

structured naming scheme for information technology systems, software, and packages

- specification, CPE dictionary, schema available
- cpe:/part:vendor:product:version:update:edition:language:sw_edition:target_sw:target_hw:other
  - part: `a` application, `h` hardware, `o` operating system
  - update: updates, service packs, etc.
  - edition: *deprecated* in 2.3
  - '*' represents ANY; '-' represents N/A
- examples (formatted string):

```
cpe:2.3:a:microsoft:exchange_server:2019:cumulative_update_5:*:*:*:*:*:*
cpe:2.3:o:canonical:ubuntu_linux:22.04:*:*:*:lts:*:*:*
cpe:2.3:a:oracle:database_server:19c:*:*:*:*:*:*:*
cpe:2.3:a:fortinet:forticlient:6.2.0:*:*:*:*:macos:*:*
```

CVE

list of entries for publicly known cybersecurity vulnerabilities

- CVE is a unique identifier for a vulnerability
  - ID number, brief description, references
- CVE Numbering Authority (CNA) – vendors and organizations authorized to assign CVE IDs
- National Vulnerability Database (NVD)
  - enhanced CVE content
  - CVSS
    - base score – computed from impact and exploitability subscores
    - impact subscore – confidentiality, integrity, availability
    - exploitability subscore – attack vector, attack complexity, priviledges required, etc.
  - CPE (affected SW), CWE

CIS Benchmark – general structure

- Level 1 and Level 2 profiles
  - sometimes for OS further distinction
  - Server/Workstation, Domain Controller/Member Server, Webserver/Proxy/Loadbalancer
- Level 1:
  - practical and prudent
  - provide a clear security benefit
  - not inhibit the utility of the technology beyond acceptable means
- Level 2:
  - intended for environments or use cases where security is paramount
  - acts as a defense in depth measure
  - may negatively inhibit the utility or performance of the technology

CIS Benchmark – NGINX (v2.0.0)

1. Installation (2 recommendations)
2. Configure Software Updates (2)
3. Minimize NGINX Modules (4)
4. Account Security (3)
5. Permissions and Ownership (4)
6. Network Configuration (4)
7. Information Disclosure (4)
8. Logging (7)
9. TLS/SSL Configuration (14)
10. Access Control (2)
11. Request Limits (5)
12. Browser Security (4)

CIS Benchmark – structure of a recommendation

- Profile Applicability
- Description
- Rationale
- Audit
- Remediation
- Default Value
- References
- CIS Controls

DISA STIG – general structure

- STIG – Security Technical Implementation Guide
- XCCDF format
  - profiles, groups, rules, etc.
- Severity Category Codes (CAT I, II, III) for vulnerabilities
  - measures a degree in loss of Confidentiality, Availability, or Integrity (CIA)
  - CAT I – direct and immediate loss of CIA
  - CAT II – a potential loss of CIA
  - CAT III – degrades measures to protect against loss of CIA
- sometimes additional and supplemental documents

Tenable audit file checks – example

- xml, tags with custom semantics
- Oracle Linux 8 (DISA STIG)

```
<custom_item>
  system          : "Linux"
  type            : FILE_CONTENT_CHECK
  description     : "OL08-00-010201 - ... all ... SSH traffic ... terminated ...
                    after 10 minutes of inactivity."

  file            : "/etc/ssh/sshd_config*"
  regex           : "^[\\s]*(?i)ClientAliveInterval(?-i)[\\s]"
  expect          : "^[\\s]*(?i)ClientAliveInterval(?-i)[\\s]+([1-9]|[1-9][0-9]|
                    [1-5][0-9]{2}|600)[\\s]*$"
  file_required   : NO
  min_occurrences : "1"
```

Exercises

1. Download and install Nessus Essential scanner or configure Qualys Community Edition. Use it to scan a vulnerable system, e.g., an old version of some Linux distribution. Perform both authenticated and unauthenticated scans. Document the results.
2. Choose any CIS Benchmark. Find a recommendation that you think is important and that could be overlooked or neglected by a system administrator. Justify your answer.

Resources

1. Core Security, *The Threat & Vulnerability Management Maturity Model*, White Paper, 2014
2. National Checklist Program Repository
3. Center for Internet Security, CIS Benchmarks