EBU

OPERATING EUROVISION AND EURORADIO

R 160s1

VULNERABILITY MANAGEMENT FOR MEDIA COMPANIES AND MEDIA SYSTEM VENDORS

Supplement 1: Security Testing Guidelines

Geneva
September 2023

Document History

EBU Committee	TC	
Drafting Group	Media Cybersecurity Group (EBU MCS)	
First published	September 2023	
Revised		

Publication Keywords: Cybersecurity, Media Systems, Vulnerability, Vendor, Vulnerabilities, CVE, CVSS, CAN, Security Testing, Ethical Hacking.

Acknowledgement

EBU technical publications are the work of experts from EBU Members, Associate Members and third parties consisting of international standards bodies, industry partners, academic institutions and independent consultants.

Their contribution to EBU technical publications is a very generous act by the individuals concerned and by their employers. The EBU appreciates their efforts and thanks them most sincerely.

This document has been produced with the assistance of the following entities:

EBU Members: Media Cybersecurity Group.

EBU Project Manager: Lucille Verbaere

Disclaimer

Note that this Supplement may be updated from time to time as experience and feedback is received from EBU Members.

Security Testing Guidelines

1. Introduction

The EBU encourages EBU Members to test their media systems and, when vulnerabilities are identified, liaise with their vendors so that they fix them, or implement mitigations if not fixable.

This document is published as a supplement to the revised R160v2 (September 2023). It provides guidelines for performing vulnerability assessments of broadcast equipment (Device under test - DuT). It outlines:

- **Basic tests** (mostly automated) that should be performed for every DuT to ensure a minimal security baseline.
- **Advanced tests** for in-depth and manual analysis of possible vulnerabilities. They should be performed if the DuTs Threat-and-Risk Assessment shows higher risk and/or is business-critical, as well as subject to available resources.

Testcase	Basic tests	Advanced tests
Passive Checks	X	
Port Scanning	X	X
Vulnerability Scanning	X	X
Web Application Scanning	X	X
Password Security	X	X
Firmware Analysis		X
Analysing Network Traffic		X
Management Interface Protection	X	X
Crypto	X	X

A triage guideline will also be developed to help assess the criticality of each vulnerability found and prioritize mitigation actions accordingly.

Note: *Testcases listed in this document may be linked to requirements of the EBU R143 Annex B (Product Security) in the format R143 <section>-<requirement> (e.g., R143 B DO-01 links to R143 Annex B DO-01)*

1.1 Vulnerability Scanning vs. Penetration testing

Vulnerability Scanning describes the use of automated software tools to identify/uncover security issues (vulnerabilities) that may affect the DuT. Automated tools test via pre-compiled attack patterns or by fuzzing inputs of an application and observing a system's/application's response.

Fully automated tools can be used to set up a continuous security scanning process to monitor DuTs for possible software vulnerabilities. This methodology can uncover security issues soon after they get introduced, **but it is only applicable for known/broadly deployed software that is covered by these tests**. For testing (non-default) custom firmware/software, manual testing (penetration testing) is required.

Additional tools for automated **application-level** tests often offer a hybrid approach that enables a penetration tester to conduct additional, guided tests. Furthermore, a penetration tester can perform manual analysis that offers a custom-tailored analysis of the security of the DuT. This process may uncover more vulnerabilities than an automated process, but it is harder to implement as a continuous process.

Usually, a penetration test should be performed initially before taking the DuT into production or after changes in the feature set (for example, after applying device upgrades).

2. Prerequisites

The following steps should be performed for a DuT before engaging in any type of testing:

- Upgrade the DuT to the latest firmware and/or software versions made available by the manufacturer, if possible (*).
- The software- and hardware-version should be documented.
- The configuration of the DuT (**default, test or production**) should be documented.

The following steps should be performed for vulnerability scanners, or for any other software used to execute the tests:

- Upgrade the scanners and the software to their latest stable versions.
- The versions of the used software and their respective plugins/feeds should be documented.

Regardless how you set up your test environment, it is most important to perform the tests. If your system has vulnerabilities and cannot be patched, tests results should be used to mitigate the risk outlined by the test, and trigger mitigation actions such as patching, isolation, etc.

(*) Note: *If any of these prerequisites cannot be implemented, for example, due to breaking production configurations (system/functional groups, protocol incompatibilities, etc.) the restrictions and resulting test setup should be documented to give a baseline for triaging. Pending security updates that fix known vulnerabilities (see vendors changelogs) should especially be taken into consideration.*

3. Testing

3.1 General instructions

If possible, the DuT should be isolated from production networks. Ideally, the tests are performed in a lab setup (back-to-back). An exception to this setup can be the case of passive sniffing of network traffic, which relies on fully connected devices.

It must be considered that vulnerability scans can influence the availability of the system for 3rd-parties. These also includes security systems/appliances present in your corporate environment. These can, conversely, influence the vulnerability scanners.

If it is not possible to conduct the tests in an isolated network environment or to isolate the DuT from a production environment (e.g., Active-Backup) the following measures are recommended for setting up a scanning policy and minimizing impact on the DuT and 3rd-party devices.

These measures should also be implemented if individual services or the entire device fail during testing. An indication of possible compromised results are error messages in the output of the scanner or an unresponsive device.

- Reduce the number of parallel checks.
- Disable checks that could disrupt services or impact the network/other components.

Be careful not to enable routing/bridging from the machine hosting the scanning software and to accidentally create connections between public and private/company networks. Ideally all network interfaces not actively used for the test on the scanning machine should be deactivated.

Depending on the DuT and type of tests, scans can take quite some time to finish.

The tests using network-based tools described in the following sections should be run against **each network interface** of the DuT. All abnormalities (e.g., unresponsiveness) that occur during testing should be documented.

3.2 Passive Checks

Prior to running actual tests against the DuT a basic reconnaissance can be carried out by gathering information about known vulnerabilities in the software components used.

Basic tests

Initial checks should contain a review of the vendor's manuals and changelogs for the DuT. The device's documentation should be investigated for (see also R143 B DO):

- Available network services (how to en-/disable, functionality).
- Default credentials (if any, can they be changed?)
- Differentiated user roles (Admin vs. user).
- Available security functions.
- Hardening guidelines.
- Possible past vulnerabilities in software changelogs and patch management.

By using the DuT's informational or status functions, some information about DUT's software components and their respective versions may be retrieved and used to look up known vulnerabilities matching this software. Information about known vulnerabilities is available via free databases or paid services.

Knowledge about possible vulnerabilities can be used to provide an initial assessment of the DuT and to design tests accordingly.

Advanced tests

Not in scope of this document (e.g., Social engineering).

3.3 Port Scan / Service Detection

The aim of port scanning and service detection is to identify the network attack surface of the DuT.

The tests should be carried out **for each network interface** (back-to-back setup if possible).

If it is not possible to run a port scan, a passive analysis (see § 3.8) can be performed to get (at least) basic information about the services used by the DuT. You can use Wireshark to identify ports on a system if an active port scan may harm your device's availability. Bear in mind that you may not be able to differentiate between the ports that the device is listening on (with a service) and the ports that may be used as a "client-port". You may also miss ports that are not transferring data during the passive analysis.

Basic tests

Before running a port scanner, it must be ensured that the DuT is reachable.

A port scanner is used to detect services available on a DuT. For a basic port scan, a DuT should be checked for all available TCP and the most common UDP ports.

For example, a basic port scan using Nmap (port scanner) can be configured as:

All TCP Ports:

```
$ nmap -sS -Pn -T4 -p- -oA <output_file> <target>
```

```
TOP 1000 UDP Ports:
```

```
$ nmap -sU -Pn -T4 –top-ports 1000 -oA <output_file> <target>
```

The services identified should be compared to the DUT's manufacturer documentation (R143 B DO-03) and differences should be documented. Each service discovered should be checked for secure communications (see § 3.10, basic tests).

Advanced tests

In addition to a plain port scan, some of the available tools also provide functions to gain more insight on a DuT and its services, such as, for example, the operating system or service detection.

For the prior Nmap example the following flags can additionally be used:

`-O`:	Perform an operating system detection
`-sV`:	Perform a service detection
`-sC`:	Script-Scan – Use scripts from different categories to gather more information about services and do basic vulnerability-checks

Note: *Additional and more thorough test may cause a higher load and thus impact the DuT and its services in availability.*

If the additional tests do impact the availability of the DuT, the following actions can be taken to reduce the possibility of failure during a scan:

- Disable the service detection (`-sV`). Open ports will be detected, but no banner information is retrieved from the services.
- Disable OS-detection (`-O`). This can reduce the number of probes sent to the DuT.
- For TCP ports, change the SYN-Scan (`-sS`) to a full-connect-scan (`-sT`)
- Reduce the timing options to a lower level (-T1, …, -T5) to slow down the scan. A higher value (more aggressive timing) may impact the accuracy of the results.

For adjustment of the corresponding options refer to the manual of the port scanner software in use.

For cross-checking a possible impact on the DuT during any invasive testing described in this document, additional port-scans can be performed. Differences in the available ports and services may indicate a service failure or compromise due to a test carried out.

3.4 Vulnerability Scan

To check a DuT for known vulnerabilities, an automated vulnerability scanner can be used. This type of software runs precompiled tests based on knowledge of existing vulnerabilities against the DuT and rates the corresponding responses by known patterns.

*Note: This type of scan relies on having the most recent software and feed versions available. It is also important to mention that this kind of test performs well on generic services but **may yield no results on custom devices that do not use "mainstream" software**. These tests should be run against **all network interfaces** of the DuT.*

The results from an automated vulnerability scan should be cross-checked manually afterwards for false positives.

Basic tests

Before performing any automated vulnerability scan, the DuT should be in a production configuration. Default configurations on the DuT may raise generic vulnerabilities caused by, for example, weak credentials. If running the vulnerability scanner against a DuT in a default configuration (e.g., during generic testing), possible measures for hardening the configuration should be derived from the results.

Basic scans don't need a fine-tuning of the scan-configuration, but should meet the following prerequisites, to get better results:

- Scan all TCP-Ports, instead of default port-range (results from a port-scan may be used here). If the scan is using a predefined default-list it may not contain all ports of interest and the results may miss out some vulnerabilities
- If possible, configure the scan to consider all targets as `Alive`.

Note 1: *Some DuTs don't respond to ICMP Pings (firewalled, or the Operating Systems network stack doesn't support it). In this case, the scanner must not perform a live-check via ICMP (which fails and thus terminates the scan process) and instead must assume the system is running and is available for a port-scan.*

Note 2: *If you are using NMAP, ICMP Ping is not to be confused with other Ping types, such as TCP-SYN, TCP-ACK and ARP Ping. TCP-SYN and TCP-ACK Pings work similarly to their corresponding port scans, but they terminate (report the result) as soon as the first port replies with an ACK. There is no such thing as TCP-Connect Ping in Nmap. Therefore, in case there is legitimate concern that the DuT may crash due to too many half-opened TCP handshakes, then TCP-SYN and TCP-ACK Pings should be avoided as alive test methods. The same logic applies for port scanning – you should use TCP Connect scan (`-sT` flag) in case you are concerned the DuT might crash. In case the DuT is on the same LAN as the scanner, we recommend using ARP Ping as an alive test method (if necessary).*

Additional to the basic vulnerability scan, many products offer additional checks to be run against the DuT. Depending on the scanner solution in use there may be options for performing basic web application security tests, authenticated scans and more.

Authenticated scans (performed on DUTs with remote-management-access such as SSH, Telnet or WMI) can yield more detailed results than a basic vulnerability scan. An authenticated scan can use the operating system tools of the DuT to enumerate installed software and their respective versions and directly check configuration files. This type of scan can also be used if the DuT becomes unresponsive during an unauthenticated vulnerability scan through its network interfaces.

Note: Automated vulnerability detection systems can cause a high load on the DuT and impact the system's availability. If there are issues during a vulnerability scan, the following actions may be taken to reduce the probability of DuT failure:

- Reduce the number of parallel tests.
- Select only `Safe Checks`.
- Don't use brute-force mechanisms (configuration scanner-dependent).
- Exclude fragile services of the DuT from the scan.
- Run an authenticated scan if possible.

Furthermore, networked services of the DuT, especially custom ones, should be investigated by passive analysis (see § 3.8) and a manual analysis and/or fuzzing (e.g., input validation) should be performed. Depending on the type of service there may be software tools available for automated testing. These additional tests can uncover flaws that may lead to an impact on confidentiality, integrity or availability.

Example for setting up a basic Greenbone scan.

Greenbone scans created via the default scan-wizard contain a port list that includes only the IANA assigned TCP ports. If the vulnerability scan must be performed on all TCP and UDP ports, the scan task must be created manually. Greenbone comes with predefined port lists containing different sets of TCP and UDP ports (Configuration/Port Lists). Additionally, custom port lists can be created. The results of the port-scan can be used to create a specific port-list for a DuT.

Furthermore, the "Alive Test"-option in the target creation dialogue should be set to "**Consider Alive**" to increase the quality of the results - if the port-scan via NMAP showed open ports, then the device is certainly alive.

When creating a new scan task, you can select the custom target with its port list and assign a scan policy. The default policy should be "**Full and Fast**".

After manually setting up a scan, don't forget to start the task. This does not happen automatically if you are not using the default wizard dialogue.

3.5 Web Application Scan

Due to their complexity, web applications usually offer the largest attack surface and, at the same time, represent the interface through which all users interact with the system. If a system offers a web application, it should be tested for possible vulnerabilities.

Basic tests

For basic security tests of a web application there are multiple scanner options to choose from. In general, the software available are either:

- Built-in in Vulnerability Scanner (product-dependent) and fully automated.
- Standalone fully automated tools.
- Hybrid/Proxy-based assisted web application security scanner.

Basic tests can use any tool from the above categories to run an automated scan but should always be configured to match the web applications properties. For example, it is important to direct the fully automated scan to start at the right path of the web server or use authentication, when applicable. Basic tests should try to cover the most common web vulnerabilities described in (R143 B AP) or OWASP Top 10 and check if the product is hardened against these types of vulnerability.

Possible tools for performing basic tests are *Nuclei* (extensible template-based web security scanner), *Nikto* (geared towards webserver security), the automated scanner components of *OWASP Zap* or *Burpsuite* (web application logic) and additionally fuzzers such as *Skipfish*.

Advanced tests

If the web application offers business critical operations and/or cannot be sufficiently tested with automatic scans (due to complexity, login mechanisms, …), a tool from the last category (e.g., assisted web application scanner such as OWASP Zap, Burpsuite …) should be used to perform manual analysis and highly customized tests adapted to the functions of a web application. Tests with this kind of tool can produce significantly better results than fully automated tests and they often support complementary testing technologies such as Websockets or hijacking of user-sessions/session-security.

Note that the tester conducting the scan must have prior knowledge of the tested components, their functionalities and their used technologies to properly test the DuT. Depending on these factors, it may be necessary to perform more intensive manual tests for vulnerabilities from the publicly available OWASP Top 10 list, as well as the following:

- Injections – SQLi, NoSQLi, XEEi, SSTi or OS command injection. Dedicated, automated tools exist for testing some of these vulnerabilities (such as *sqlmap* for SQLi and NoSQLi) that can be used in conjunction with manual testing or the web application scanner. These tools require prior knowledge of the tester.
- File inclusions and directory traversals.
- Authentication and Access-Control issues and bypasses – the tester should at least check if the issued tokens are indeed necessary for accessing the DuT's API (and not just the UI), or if the tokens contain simple control flags that can be manipulated client-side (for example, an unprotected "is-admin" flag in the token). Depending on the authentication algorithm being used, more advanced and specific tests may be necessary (for example, in case of JWT-based authentication, an extensive security assessment should contain checks for vulnerabilities such as JWT header injection and JWT algorithm confusion).
- Unrestricted file uploads – check if the server allows arbitrary types of files to be uploaded and if they are stored outside the server's static resource folder. If this is the case, RCE via uploaded code may be possible. Prior tester knowledge on how to masquerade the actual file type during upload is required.
- Logic bugs – this category encompasses a wide range of vulnerabilities that arise due to inherently flawed assumptions developers make about the ways users will interact with the application. They are impossible to detect with automated scanners and require the tester to be fully acquainted with the functionality of the web service.
- Other advanced vulnerabilities such as insecure deserialization, HTTP request smuggling, WebSocket and JWT attacks etc.

Note: *Manual tests performed in this category can also be used to assist in verifying results from previous automated vulnerability scans, especially for those results that have a lower "quality of detection" (in other words, for results for which the automated scanner could not tell with absolute certainty whether the vulnerability is present). For example, an automated scanner may report a possible HTTP request smuggling* *vulnerability in a module of the identified web server simply based on the version fingerprint of the server. If a backport patch has been applied or if the DuT does not make use of the vulnerable module, the vulnerability does not affect the DuT. Manual checks are required to verify this.*

3.6 Password Security

All available network services should be checked for default or hardcoded passwords. It is also necessary to check if passwords can be changed by the user.

Basic tests

Basic tests for testing against weak passwords include (see also R143 B AA):

- Checking the documentation for default passwords.
- Checking the documentation for how to change passwords.
- Performing a password change for all services and accounts (verification).
- Checking if one password is valid for more than one service/user/role on the DuT.
- Checking if the device implements a password policy.
- Checking if the device supports MFA (Multi-Factor-Authentication).

Additional active checks for weak passwords can be performed by automated brute-forcing tools that rely on pre-assembled wordlists or dictionaries. The wordlists should contain the most common passwords (freely available) and/or passwords up to a given length.

It is important to choose a brute-forcing tool based on the service and the underlying authentication protocol being used. There are many versatile tools that support testing multiple protocols (*ncrack*, *hydra*, *medusa* etc.).

Note: *Some services may throttle connections if they detect brute-force attacks. Adjust the tests accordingly.*

Furthermore, it should be tested if the product implements mechanisms to protect against account enumeration. This includes checking for overly descriptive error messages that may contain information that can be used to gain additional knowledge about the DuT's internals.

Advanced tests

Advanced tests in this category can be found in the sections **Crypto** and **Firmware Analysis**.

3.7 Firmware Analysis

Firmware analysis can be performed to gain a deeper insight into the product's functionality and implementation. Possible outcomes of these type of tests include the following:

- Detection of hardcoded passwords, credentials and authentication backdoors (see R143 B AA)
- Detection of insecure configurations or vulnerabilities in the source code (if available, or reverse-engineered).

Basic tests

This category of tests does not contain basic testing.

Advanced tests

Every firmware analysis procedure should begin with a comprehensive scan of the firmware archive for known file signatures. File signatures are special byte sequences (also known as magic numbers) placed in file headers that identify the type of the file being used (for example, an ELF64 binary, TAR archive, etc.). The most used tool for this initial step is *binwalk*. It is important to understand that signature checking can lead to false positives – certain file types have short signatures, which increases the probability of the occurrence of the magic number somewhere in the firmware archive, although the corresponding file type is not present.

The previous procedure may detect misplaced ZIP, TAR or other compressed archives withing the firmware archive that contain source code or other sensitive information. In this case, the tester should review the code offline and use the acquired information to guide further testing of the DuT.

More commonly, the firmware archive will contain only pre-compiled binaries. In this case, it is still possible to perform reverse engineering using *decompiler* and disassembler tools such as *IDA*, *Ghidra*, *BinaryNinja* or *radare2*, although this would require a lot of time and is typically legally forbidden due to license agreements. A viable alternative would be to use hex editor and viewer tools (such as *xxd* and *strings* utilities) to search for hardcoded passwords and credentials directly within the original binary.

Moreover, if access is possible, an analysis of the running firmware should be performed (via an SSH connection to the device for example). In addition to the offline tests, this analysis should include additional tests that check in which user context the services offered by the system are running. For example, one could check if a certain network service is running with the least number of privileges possible, or if the operating system, kernel and system libraries have been patched against known local privilege escalation bugs etc.

3.8 Analysis of Network Traffic

Analysing network by passive monitoring traffic of a DuT can yield some insights into the protocols used and possible vulnerabilities such as cleartext-transmission of credentials.

Basic tests

This category of tests does not contain basic testing.

Advanced tests

Analysing network traffic originating from and going to the DuT usually requires a basic lab-setup that mimics a minimal production environment that the device is meant to run in. It may be possible to use back-to-back setups for simple applications, but this may not cover all the possible communications supported by the DuT. If applicable, a production environment can be used for this kind of test, as it is non-invasive.

Tests in this category can check for:

- Cleartext credentials communicated via network.
- Cleartext protocols and their functionality.
- Undocumented traffic originating from the device.
- Undocumented traffic going to the device (see R143 B NE-02).

Results from this analysis can be reused as information for other tests, such as, for example, forensic analysis or password security (see § 3.6 & § 3.10).

Recorded communications can be used for creating a communication matrix for the device and discovering services active on the DuT. This may be useful in scenarios where a direct port scan is not feasible for different reasons. Note that this method is not a comprehensive replacement for port scanning, as certain network services may not be active during the test. Moreover, it may be difficult to differentiate between server and client ports.

The most used tool for passive network analysis is *Wireshark*.

Furthermore, these results can be used to create fuzzing attacks to check for possible vulnerabilities in the protocol used by the service of the DuT.

3.9 Management Interfaces (/ Separation of Network Interfaces)

Many devices offer dedicated network interfaces for management purposes. It should be tested if the services needed for managing a DuT are limited to the dedicated management interfaces and are not present otherwise/network traffic is not routed between interfaces.

Basic tests

Based on the results of the port-scan for every networked interface (wired or wireless) of the DuT, check which services are available and if there are management functions available on non-management interfaces (see also R143 B NE-01)

If management services are available on non-management interfaces, it should be checked if it is possible to limit the listen-address of the service in question to the appropriate interface (check the vendor's documentation).

Advanced tests

Test routing of network traffic between the DuT's interfaces. For example, check whether traffic sent to a management interface can be routed through the DuT to a production-only network segment and vice-versa.

The design of these tests should be based on the planned deployment configuration of the DuT to gain meaningful results. As with the passive network traffic analysis, this test needs the DuT to be run in a setup as similar as possible to the production setup.

3.10 Cryptography / Encryption

Basic tests

Results from an automated vulnerability scan should be checked for possible issues regarding protocols that are implementing cryptographic functions/methods or that are missing them:

- unencrypted services, like HTTP, Telnet, FTP, LDAP, SNMPv1/2 etc.
- services with weak encryption (SSLv2, SSLv3, TLS below version 1.2, TLS and SSH with weak cipher suites etc.)
- see R143 EN-03 recommended protocols.

If the DuT uses cleartext or insecure protocols, its documentation should be checked to see if they can be replaced by encrypted or secure alternatives.

Advanced tests

Using specialized tools, additional tests can be performed for checking the configuration of cryptographic protocols used by the DuT, as well as possible issues regarding user credentials and secrets gained by other tests described in this document.

Protocol-specific tools can be used to check if the service running on the DuT is configured to use acceptable cryptography. For example, *sslscan* can be used to externally assess the TLS configuration of the service. Results from these tools can also help implement a secure configuration for the service concerned.

If the DuT offers MFA (Multi-Factor-Authentication), it should be checked whether the used mechanism is considered secure at the time of testing. Additionally, the effectiveness of the MFA-mechanism used should be proven by tests. See R143 B AA-07.

Some of the tests described in this document, such as checking the firmware for hardcoded secrets or passive analysis of network traffic, may produce hashes. It should be checked if the type of hash being used is weaker than the current state-of-the-art hash-types (see R143 B EN-01, EN-02, EN-05).

Furthermore, similar to brute-force tests described in § 3.6, it should be checked if the obtained hash was produced by a commonly used (weak) secret. Popular tools for brute-forcing hashes are *john* and *hashcat*.

Note: *These tests may take a lot of time and heavily depend on the quality of the used wordlist/dictionary.*