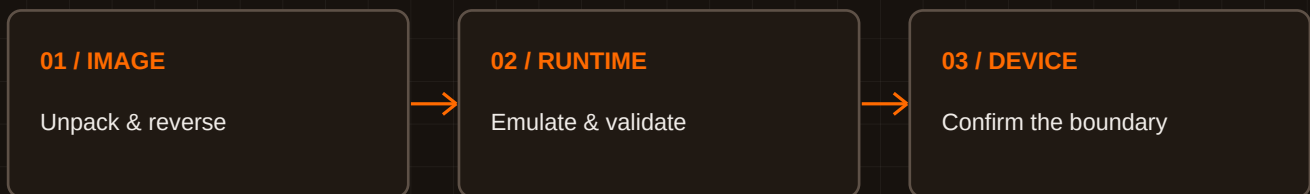


FIRMWARE / EMBEDDED SECURITY - SYNTHETIC SAMPLE

Firmware & Embedded Penetration Test

From the image to the boundary that matters.



Illustrative assessment workflow - not a captured test session

A fictional laboratory scenario. This document illustrates the depth, structure and evidence discipline of a firmware deliverable. The product, findings, commands, captures and outcomes are synthetic. No assessment was executed to produce this sample; no real customer or device vendor is represented.

5 illustrative findings / 3 High / 2 Medium
Scope, reproduction, remediation and retest criteria

Prepared by PXLSec
PXLSecurity LTD. / Sofia, Bulgaria
Version 1.0 / 28 October 2026

SECTION READ FIRST

How to read this sample

The layout follows the existing PXLSec report family. The technical story is entirely fictional.

Document	PXLSec-FW-SAMPLE / v1.0 / public marketing sample
Product	SG-LAB gateway - fictional product, not a vendor model
Assessment dates	Not applicable. No actual engagement or test was performed.
Evidence status	All transcripts, files, results and observations are illustrative.
Remediation status	Not retested. Criteria are a plan, not a closure claim.
Classification	Public sample. Real customer reports follow agreed handling rules.

Reading route

Start with the executive summary and scope. The coverage matrix states which conclusions require an image, an execution environment or a real device. Each finding then gives a starting point, the illustrated proof, explicit limits, remediation and measurable retest criteria.

Section	Pages
Executive summary	3
Scope and threat model	4
Method and coverage	5
Five detailed findings	6-15
Remediation roadmap	16
Retest plan	17
Limitations and references	18

Evidence discipline. A scanner match is a candidate. Emulated behaviour is a result in that environment. A physical-device result applies to the stated unit and configuration. This sample keeps those labels separate.

SECTION 01

Executive summary

The sample shows how recoverable secrets, update trust and privileged interfaces can undermine a connected product.

5 illustrative findings	3 High	2 Medium	0 real clients represented
-----------------------------------	------------------	--------------------	--------------------------------------

The fictional SG-LAB image exposes a shared credential that a laboratory API accepts across two device identities. Independently, the application updater accepts a modified unsigned payload, and an authenticated diagnostic function invokes a shell in a root process. Those are different trust failures: identity scope, update authenticity and input interpretation.

Two further examples concern the agreed minimum-version policy and a production-mode UART console. Their severity depends on the stated administrator and physical-access prerequisites. Neither is presented as an unauthenticated Internet compromise or a universal property of embedded devices.



■ High 3 ■ Medium 2

Distribution of the five synthetic findings only. Not a benchmark of tested products.

Decisions before the next release

Retire the shared authority and enforce device-level server authorisation. Require authenticated update payloads. Remove shell interpolation from the diagnostic path. Then align rollback and maintenance behaviour with the actual product requirements.

No certification or absence-of-vulnerabilities claim. This sample is not a compliance assessment, a CRA conformity statement or proof that an actual device has these defects. A real engagement is bounded by its targets, access, techniques and test window.

SECTION 02

Scope and attacker assumptions

A finding is meaningful only with the product state, access and environment that make it possible.

Units	SG-LAB-A and SG-LAB-B; two fictional production-mode lab units
Builds	Application image 1.4.2 and older signed lab package 1.3.9
Architecture	Illustrative ARMv7 embedded Linux userspace; not a platform support guarantee
Image identifier	Synthetic manifest identifier, not a hash of an actual firmware image: 995dd0d6419a1ddca4ab958aaa7f585f4ce8e3a733a7067a5e0e6179ef49e1d4
Execution	QEMU process environment for a selected management daemon; device confirmation shown separately
Network	Isolated management address 192.0.2.10; fleet.example.test for the authorised lab API
Accounts	Unauthenticated image holder; operator; update administrator; physical device holder
Source code	No source code assumed; pseudocode reconstructs the synthetic behaviour

In scope

The supplied image, application updater, diagnostic feature, production serial console and the isolated device-state API. Physical access assumes permission to open the enclosure and use identified interfaces. Invasive extraction, destructive tests and electrical fault injection are not included in this fictional scope.

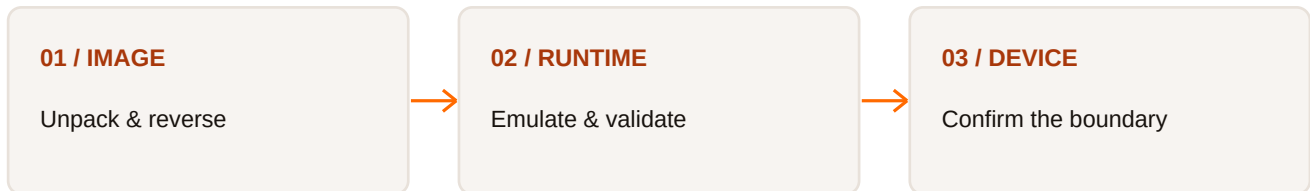
Out of scope

Production fleets, customer data, unrelated cloud services, radio-protocol assessment, silicon decapsulation, side-channel work and a ROM/root-of-trust bypass. Named tools and protocols are methods, not a claim that all architectures or peripherals can be emulated.

SECTION 03

Method and coverage

Coverage records evidence provenance, not only a checklist of tool names.



Illustrative assessment workflow - not a captured test session

Area	Illustrated method	Result and limit
Image and provisioning	Unpack + compare supplied artefacts	Shared private material; identity value omitted. FW-01.
Fleet authorisation	Positive and negative laboratory API requests	Cross-device read only, no production or fleet-wide test. FW-01.
Update authenticity	Package modification + device update	Application payload accepted; not ROM secure-boot bypass. FW-02.
Diagnostic execution	Binary trace + QEMU + device check	Operator-to-root command execution. FW-03.
Rollback policy	Compare requirement and installed build	Policy floor bypass; no historical CVE exploit. FW-04.
Physical console	Production-mode UART interaction	Unauthenticated root with physical prerequisites. FW-05.
Other components	Inventory / targeted triage only	No claim of exhaustive binary, CVE or protocol coverage.
Hardware root of trust	Not exercised	No conclusion about fuses, ROM checks or secure elements.

Validation rules

In a real engagement, record artefact hashes, environment configuration, account/role, command or request, response and negative controls. Preserve only evidence necessary to substantiate the conclusion; recovered private material is handled separately through the agreed secure channel.

Severity model

High and Medium are contextual illustrative ratings for the stated scenarios. Numeric CVSS scores are intentionally not invented. A real report records the agreed scoring version, vector and environmental assumptions where applicable.

SECTION FW-01

A shared firmware credential permits cross-device API access

Illustrative finding - synthetic evidence, not an executed assessment.

HIGH

Component	Provisioning material and laboratory fleet API
Starting access	Possession of the supplied firmware image; network access to the lab API
Evidence provenance	Static image + isolated laboratory API (synthetic)
Classification	CWE-798 / CWE-862
Status	Illustrative / remediation not retested

Description

The fictional image contains a client certificate and its private key in the default provisioning directory. In the sample scenario, that credential is identical in the two supplied lab units. The laboratory API accepts it as a product-wide principal and does not restrict the requested device identifier to a separately authenticated device identity.

Demonstrated impact in this scenario

A party holding this image can read the status of the second in-scope lab device. This demonstrates cross-device information access in the sample environment. Access to customer data, arbitrary fleets, command execution and production services is not demonstrated or implied.

EVIDENCE A / SYNTHETIC ARTEFACT OR REQUIREMENT

```
$ lab-inspect image-1.4.2.bin --path /etc/sg/provisioning/
client.crt client.key api.conf
$ lab-inspect compare-cert unit-a unit-b
certificate fingerprint: identical in both supplied images
private key: present (value deliberately omitted)
```

EVIDENCE B / SYNTHETIC VALIDATION

```
$ lab-api-check --identity recovered-lab-client \
--device sg-lab-b --operation read-state
HTTP 200
{"device_id":"sg-lab-b","state":"lab-idle"}
$ lab-api-check --identity none --device sg-lab-b
HTTP 401
```

This example proves only the authorised laboratory read-state operation. The transcript uses fictional helper commands, not a real executed tool session. No live private key, token or customer record is included.

SECTION FW-01 / CONTINUED

Reproduction, repair and verification

The follow-through matters as much as the initial observation.

Root cause

A shared identity is shipped as recoverable client material, and the server treats successful authentication as sufficient authority for device-specific data. Encryption of the image alone would not repair that server-side authorisation boundary.

Reproduction approach

- 1 Identify provisioning files in the supplied image and document their build and path. Do not publish recovered private material in the report.
- 2 Compare the public-key identity across the two authorised lab units; record the matching fingerprint as illustrative evidence.
- 3 Authenticate to the isolated API with the recovered test identity and request the second agreed device identifier.
- 4 Repeat without a credential as a negative control. Preserve the minimal response needed to establish cross-device access.

Remediation

- Revoke or replace the exposed shared credential and assess available evidence of prior use. Plan migration so old devices do not silently retain authority.
- Issue unique device identities through a controlled provisioning process. Restrict private-key access using a design appropriate to the hardware and threat model.
- Authorise every device-specific server operation against the authenticated principal. A valid client certificate is not itself permission to read any device.
- Keep test/provisioning credentials out of release images and add release checks for accidental inclusion.

Retest acceptance criteria - not yet performed

Confirm the old shared identity is rejected; device A can read only its allowed resources; requests for device B are denied; and a new image does not reintroduce the shared private material. Test the server boundary even if storage protection changes.

Residual risk and boundary. Unique keys reduce shared-identity exposure but do not prevent misuse of a compromised individual device. The server must still apply narrow permissions, revocation and monitoring.

SECTION FW-02

The update agent accepts an unsigned application payload

Illustrative finding - synthetic evidence, not an executed assessment.

HIGH

Component	Application update package verification
Starting access	Authenticated operator on the in-scope management interface
Evidence provenance	Static update logic + device update exercise (synthetic)
Classification	CWE-347 / CWE-494
Status	Illustrative / remediation not retested

Description

The sample update agent checks a SHA-256 digest from a manifest included in the same archive, but accepts a package without a vendor signature. An operator can replace an application file, recompute the digest and install the modified package through the intended update workflow.

Demonstrated impact in this scenario

In the fictional exercise, an authorised test operator installs a harmless modified startup component and observes a marker after reboot. This demonstrates execution of unauthorised application content on the lab unit under the component's configured privileges. It does not demonstrate a bypass of ROM secure boot, bootloader verification or a hardware root of trust.

EVIDENCE A / SYNTHETIC ARTEFACT OR REQUIREMENT

```
# Simplified, synthetic updater logic
expected = manifest["payload_sha256"]
if sha256(payload) != expected: reject()
# No required signature verification before installation
install_application_payload(payload)
```

EVIDENCE B / SYNTHETIC VALIDATION

```
$ lab-update-check --package modified-lab-update.tar
payload hash: matches supplied manifest
vendor signature: absent
update result: accepted
$ lab-console --read /var/run/firmware-sample-marker
lab-payload-executed
boot-chain bypass: NOT TESTED
```

A valid operator session and update permissions are prerequisites. The lab marker is a controlled, non-persistent illustration. No destructive payload or production update is represented.

SECTION FW-02 / CONTINUED

Reproduction, repair and verification

The follow-through matters as much as the initial observation.

Root cause

The integrity digest detects accidental corruption but is not anchored in an independent trust decision. The person modifying the payload can modify the manifest as well. The application updater and the platform boot chain must be assessed as separate controls.

Reproduction approach

- 1 Start from the agreed application update package; preserve an original copy and its hash.
- 2 Replace a benign marker-producing component in the isolated lab payload and regenerate the package digest.
- 3 Submit the modified package without a vendor signature using the authorised operator workflow.
- 4 Observe the application marker after the approved reboot. Record the component account and distinguish that result from the untested boot chain.

Remediation

- Require a valid signature over the payload and all security-relevant metadata before installation or extraction to privileged paths.
- Anchor verification to trusted keys and algorithms; reject absent, malformed, wrong-key and invalid signatures without a permissive fallback.
- Protect verification keys and the verification path according to the device threat model. Define a recovery mechanism that preserves the same trust requirements.
- Validate archive paths and extraction behaviour independently. Add negative tests for modified payloads, metadata, signatures and key identifiers.

Retest acceptance criteria - not yet performed

A legitimate package installs; unsigned, altered and wrong-key packages are rejected before changes occur; failed verification leaves the device in a known usable state. Repeat on the real target build. Assess secure boot separately if included.

Residual risk and boundary. Correct signing does not enforce version freshness by itself. A valid but disallowed older package remains a separate rollback-policy question (FW-04).

SECTION FW-03

A diagnostic parameter reaches a root shell command

Illustrative finding - synthetic evidence, not an executed assessment.

HIGH

Component	Authenticated network diagnostic endpoint in sgd
Starting access	Authenticated low-privilege operator with diagnostic access
Evidence provenance	Binary tracing + QEMU process emulation + device confirmation (synthetic)
Classification	CWE-78
Status	Illustrative / remediation not retested

Description

The fictional management daemon concatenates the diagnostic host parameter into a command string passed to a shell. The server permits an operator to run diagnostics, but the parameter is not constrained to a host value and shell metacharacters retain their meaning.

Demonstrated impact in this scenario

A harmless identity command produces `uid=0` in the emulated process and on the in-scope lab unit. In this example, an operator can cross from a constrained diagnostic function into root command execution. Fleet access, persistence, data exfiltration and movement to unrelated networks are not exercised.

EVIDENCE A / SYNTHETIC ARTEFACT OR REQUIREMENT

```
# Decompiler-style pseudocode; illustrative only
command = "ping -c 1 " + request.host
result = shell_exec(command)
# daemon account in the sample service definition: root
```

EVIDENCE B / SYNTHETIC VALIDATION

```
POST /api/diagnostics/ping [authenticated lab operator]
{"host":"127.0.0.1; id"}
QEMU process trace: uid=0(root) gid=0(root)
SG-LAB-A response: uid=0(root) gid=0(root)
Unauthenticated control request: HTTP 401
```

QEMU alone would establish behaviour only in that execution environment. The fictional device confirmation is stated separately. The endpoint is not represented as unauthenticated, and no target outside the lab is addressed.

SECTION FW-03 / CONTINUED

Reproduction, repair and verification

The follow-through matters as much as the initial observation.

Root cause

Untrusted input is interpreted as shell syntax inside a privileged process. The role check authorises access to the feature but does not make the constructed command safe. Running the daemon as root increases the impact.

Reproduction approach

- 1 Map the authenticated diagnostic handler to the binary routine that constructs and executes the command.
- 2 Reproduce the handler in an appropriate emulated process environment, recording substituted peripherals, libraries and configuration.
- 3 Send a loopback diagnostic value with a harmless command separator and identity check using the agreed operator account.
- 4 Repeat the minimal check on the lab device. Compare results with a normal host value and an unauthenticated request.

Remediation

- Avoid a shell: execute the intended diagnostic binary with a fixed argument vector and a strictly parsed address or hostname.
- Apply semantic validation and reject additional command syntax. Escaping alone is not the preferred repair for a feature that needs only a host argument.
- Run the management service with the least privileges needed and isolate privileged operations behind a narrow interface.
- Add role-specific negative tests and confirm any process containment actually limits the diagnostic operation.

Retest acceptance criteria - not yet performed

The original metacharacter input and relevant encoding variants are rejected or handled strictly as data. Valid diagnostics still work. Confirm the production handler uses no shell interpolation and verify the effective service account on the fixed device.

Residual risk and boundary. Removing shell interpretation closes this path, but does not prove that all native parsers or other diagnostic handlers are safe. Broader input testing remains bounded by the agreed time and targets.

SECTION FW-04

An older signed build bypasses the agreed minimum-version policy

Illustrative finding - synthetic evidence, not an executed assessment.

MEDIUM

Component	Update version policy and recovery behaviour
Starting access	Administrator allowed to install packages; access to an older signed lab build
Evidence provenance	Device update and version-state comparison (synthetic)
Classification	CWE-693
Status	Illustrative / remediation not retested

Description

The synthetic product requirement sets a minimum permitted application version after a security update. The laboratory device accepts the older, correctly signed 1.3.9 build even after 1.4.2 has established the minimum-version state. Signature verification succeeds, but the independent freshness policy is not enforced.

Demonstrated impact in this scenario

An authorised administrator can restore an application build excluded by the stated product policy. The example confirms a rollback-policy failure. It does not claim that every downgrade is a vulnerability, or that a particular historical CVE becomes exploitable: no vulnerable historical path is exercised in this sample.

EVIDENCE A / SYNTHETIC ARTEFACT OR REQUIREMENT

```
Synthetic product requirement SG-UPDATE-07:  
After activation of security baseline 1.4.2,  
application builds below 1.4.0 must be rejected.  
Authorised recovery exception: none for the tested state.
```

EVIDENCE B / SYNTHETIC VALIDATION

```
$ lab-update-check --package signed-1.3.9.tar  
signature: valid  
minimum permitted application build: 1.4.0  
installation: accepted  
$ lab-console --read-version  
application: 1.3.9  
historical vulnerability exploitation: not attempted
```

The security relevance rests on an explicit fictional minimum-version requirement. A supported, intentional recovery downgrade with equivalent safeguards would require a different assessment.

SECTION FW-04 / CONTINUED

Reproduction, repair and verification

The follow-through matters as much as the initial observation.

Root cause

The updater treats an authentic package as an allowed package without checking a trusted version/security-counter policy. Recovery and normal-update paths do not share an enforced minimum.

Reproduction approach

- 1 Record the product's minimum-version requirement and any authorised recovery exceptions before testing.
- 2 Establish the configured baseline on the lab device, then supply the agreed older signed package.
- 3 Verify the signature result, installation decision and effective running version independently.
- 4 Compare normal and recovery paths where included; do not infer a hardware rollback failure from an application-level observation.

Remediation

- Define the permitted rollback/recovery policy explicitly, including who can invoke exceptions and how they are authorised.
- Enforce signed version or security-counter metadata against trusted device state before installation.
- Apply the rule to all in-scope update and recovery paths, with a failure-safe and serviceable recovery design.
- Treat counter storage and hardware enforcement as target-dependent engineering decisions, not a universal configuration checkbox.

Retest acceptance criteria - not yet performed

Both normal and agreed recovery paths reject disallowed older builds while accepting permitted current updates. Verify exceptions require the documented authority. Where a historical security fix motivates the floor, separately test the original vulnerability if it is in scope.

Residual risk and boundary. An authentic update can still contain vulnerable application code. Signing and rollback controls establish provenance and policy, not the absence of software weaknesses.

SECTION FW-05

The production-mode UART console grants an unauthenticated root shell

Illustrative finding - synthetic evidence, not an executed assessment.

MEDIUM

Component	Boot console and serial login configuration
Starting access	Physical possession, opening the enclosure and access to UART test pads
Evidence provenance	Physical interface and device console (synthetic)
Classification	CWE-306
Status	Illustrative / remediation not retested

Description

The fictional unit exposes UART pads inside an ordinary removable enclosure. With the documented lab voltage and serial parameters, the production-mode boot sequence starts a root shell without authentication. No readout-protection bypass or chip-off technique is needed for this example.

Demonstrated impact in this scenario

A person with the required physical access obtains root-level control of the supplied lab unit. Local files and configuration become accessible according to that account. Impact beyond that unit, extraction from a secure element and compromise of a fleet are not claimed. FW-01 separately illustrates the server-side consequence of a shared credential.

EVIDENCE A / SYNTHETIC ARTEFACT OR REQUIREMENT

```
Synthetic production service configuration:
console = ttyS0
serial-getty = disabled
maintenance-shell = /bin/sh
maintenance-shell-authentication = none
```

EVIDENCE B / SYNTHETIC VALIDATION

```
[illustrative UART capture from SG-LAB-A]
boot mode: production
maintenance console ready
# id
uid=0(root) gid=0(root)
# cat /etc/sg/build-id
sg-lab-1.4.2
No password or authenticated service token requested.
```

The result assumes physical possession and access to the internal pads. An exposed port alone would not prove this finding. Electrical probing and console output are synthetic; no real pinout, board or vendor is represented.

SECTION FW-05 / CONTINUED

Reproduction, repair and verification

The follow-through matters as much as the initial observation.

Root cause

A development convenience remains active in the production configuration. Physical access is assumed in the sample threat model; the product requirement calls for authenticated maintenance rather than an unauthenticated debug shell.

Reproduction approach

- 1 Confirm that the supplied unit represents the intended production state and record the physical access assumption.
- 2 Identify the authorised serial interface and verify electrical characteristics before connection; do not assume a universal voltage.
- 3 Observe boot and interact only within the approved console window.
- 4 Use an identity command and a non-sensitive build marker to show the granted authority. Record authentication behaviour and prerequisites.

Remediation

- Remove the unauthenticated development shell from production builds and enforce an explicit manufacturing/release-state transition.
- If maintenance is required, use an authenticated, constrained and revocable access design appropriate to the hardware.
- Review console services, boot interruption and recovery behaviour together; disabling visible boot messages alone is not sufficient.
- Include production-like hardware in release checks. Document how support and recovery remain possible after debug restrictions are applied.

Retest acceptance criteria - not yet performed

Cold-boot a production-mode unit and repeat the same physical starting conditions. No unauthenticated privileged shell should be available. Verify the approved maintenance route, account restrictions and recovery procedure remain functional.

Residual risk and boundary. Closing UART does not prove flash confidentiality or resistance to invasive attacks. Readout protection, secure elements and fault injection need their own evidence and scope.

SECTION 04

Remediation roadmap

An illustrative engineering sequence. Owners and dates are assigned in a real engagement, not invented here.

Priority	Action	Closure evidence
Contain shared authority	Revoke shared identity; restrict server device access; plan migration. FW-01.	Old credential rejected; cross-device request denied.
Repair update trust	Require authenticated payload and metadata with fail-closed behaviour. FW-02.	Unsigned, changed and wrong-key packages rejected.
Remove shell interpretation	Use a fixed executable and parsed argument vector; reduce privileges. FW-03.	Original injection no longer executes; valid diagnostic works.
Enforce release policy	Protect minimum-version state and authorised recovery rules. FW-04.	Disallowed build rejected in tested update/recovery paths.
Harden production maintenance	Remove unauthenticated console; preserve controlled support access. FW-05.	Cold-boot physical test shows no unauthenticated privileged shell.

Changes that must be coordinated

Identity rotation can interrupt device service if the transition is not planned. Update verification, version policy and recovery must be tested together so that a rejected update does not leave a device unusable. Firmware hardening and server permissions require different engineering owners.

Release evidence pack

In a real delivery, attach affected and corrected build identifiers, the relevant configuration, reproducible minimal requests, remediation decisions and retest outcomes. Keep raw keys, full memory dumps and customer data out of a broadly distributed report.

Release assurance. A recurring engagement begins with an agreed baseline, then reviews changed artefacts and prioritises targeted testing. It defines product families, release allowance and included effort. It is not unlimited research or a guarantee that each build is safe.

SECTION 05

Retest plan and closure rules

No fixes were supplied or retested for this fictional sample. All entries remain NOT RETESTED.

Finding	Minimum confirmation	Sample status
FW-01	Reject old shared identity and cross-device reads; preserve authorised per-device operation.	NOT RETESTED
FW-02	Reject unsigned/modified/wrong-key payloads before installation; confirm safe failure.	NOT RETESTED
FW-03	Original payload and relevant variants treated as data or rejected; valid diagnostics preserved.	NOT RETESTED
FW-04	Reject disallowed older builds in every in-scope path; document recovery exceptions.	NOT RETESTED
FW-05	Cold-boot console and maintenance tests on the fixed production-mode unit.	NOT RETESTED

Closed

Repeat the original starting conditions on the identified fixed build and show that the reported boundary now holds. Verify a relevant legitimate operation still works. Record evidence and limitations, not only a developer statement.

Partially resolved or open

Use partially resolved when an original path is blocked but the same root cause or an agreed variant remains. Keep a finding open when the demonstrated path still works. An unavailable unit or inaccessible environment is not a passed retest.

New version or new product

Substantial changes, new interfaces or different hardware may need additional scope. A successful retest of five findings does not repeat every activity from the initial engagement or attest to the entire product.

Emulation is not a closure shortcut. If the original impact depended on a device, a changed emulation result alone does not establish that the hardware build is fixed.

SECTION APPENDIX

Limitations and technical handoff

A sample should show what a report can support, and what it cannot.

Synthetic evidence

Every product identifier, transcript and outcome in this document is constructed for demonstration. Fictional lab-* helper commands describe operations rather than distributable tools. A reader cannot use this sample as a vulnerability disclosure, customer reference or proof of actual testing.

What a real engagement would add

The authorisation and rules of engagement, exact target inventory, physical-device identifiers, real artefact hashes, build/configuration provenance, dates, tester identity, actual evidence references, agreed scoring and securely delivered reproduction artefacts.

Limits of a point-in-time assessment

Testing can establish specific weaknesses and control behaviour under stated conditions. It cannot prove that a product has no vulnerabilities. Source access, device revisions, encrypted components, emulation fidelity, time and safety restrictions all affect coverage.

Reference vocabulary

CWE identifiers classify weakness patterns; they are not certifications. QEMU describes an execution environment, not proof of hardware equivalence. SBOM/component inventory supports investigation but does not by itself establish exploitability. Framework mapping, if commissioned, is technical evidence rather than a conformity certificate.

CWE catalogue	https://cwe.mitre.org/
QEMU documentation	https://www.qemu.org/docs/master/
PXLSec service	https://pxlsec.com/services/firmware-embedded-penetration-testing
Secure scoping	hello@pxlsec.com / NDA and secure transfer arranged before sensitive artefacts

Prepared by PXLSec - PXLSecurity LTD.

12 Vasil Levski Blvd., 1142 Sofia, Bulgaria

Public synthetic sample / 28 October 2026 / v1.0

No actual client, vendor, production target or secret is represented.