Medical Device Cybersecurity Testing Methodology

Cybersecurity impacts safety and effectiveness of medical devices. All systems fail; all software has flaws. Connectivity creates exposure to accidents and adversaries that can trigger these flaws. Stratigos Security's medical device cybersecurity testing approach (including vulnerability and penetration testing) seeks to identify, contextualize, and assess impact of these flaws on the potential safety and effectiveness of the system.

Where Stratigos' cybersecurity test report findings focus on the effects that impact safety and effectiveness of the device and components in scope, this document details the methods and techniques used to produce those effects. Further, it aims to demystify core concepts around dynamically evaluating device protections, serve as a reference for evaluating the comprehensiveness of testing scope and methodology, and provide additional detail on the rigor with which testing is performed for stakeholders like compliance leads, R&D teams, and regulatory reviewers.

Each section details how the method is performed, applicable FDA guidance areas addressed, sample tools leveraged, expected outcomes, and limitations. We also provide best practices around independence, rigor, and transparency of assessments.

About Stratigos Security

Founded in 2012, Stratigos Security provides cybersecurity strategic advisory, program development, risk assessment, and technical evaluation for organizations of every size, around the globe.

Stratigos has deep technical, regulatory, and policy experience in healthcare. We have helped dozens of medical device makers improve their security through commercial, non-profit, and pro bono engagement. Our founder was named Entrepreneur in Residence with the FDA, advised on creation of both premarket and postmarket cybersecurity guidance, and founded the Biohacking Village: Device Lab at DEF CON. Our consultants are recognized in the field for their cybersecurity expertise, including hardware, software, firmware, embedded systems, and – of course – medical devices.

Stratigos has performed technical testing on dozens of medical devices in the past five years and draws on expertise from more than a decade of collaboration with global regulators, medical device manufacturers, patients, physicians, public policymakers and health providers.

We aim to ensure that our approach to cybersecurity testing - including scoping process, testing protocols, communication of results, reporting, and advisory assistance - sets a high water mark for what companies and regulators should expect. Our methodology is comprehensive and aligned with FDA guidance helping safer, more effective devices reach the market without undue delay and cost.

Discovery & Reconnaissance

This phase aims to map the device's exposure surface, revealing architecture, components, data flows, and potential entry points. This enables cybersecurity testers to thoroughly analyze the attack surface and focus subsequent testing on high risk areas. Key discovery and reconnaissance activities include:

Asset Identification - Catalog relevant hardware, software, and interfaces that comprise the medical device system, supporting infrastructure, ancillary systems with data connections. Identify manufacturer, software/firmware versions, hardware model numbers.

Network Topology Mapping - Use active and passive methods to identify servers, subnet segmentation, network hardware, directionality of connections, and open ports.

Services Enumeration - Leverage banners from open ports and protocol fingerprinting to inventory running services on assets, client-server relationships, libraries/frameworks in use.

Data Flow Analysis - Packet sniffing and inspection to extract information about the system, protocols used, payload contents, frequencies, and volumes of network traffic.

Vulnerability Scanning - Use automated scanners to assess assets for known vulnerabilities such as unpatched software, risky configurations, common exposures.

Tool Examples

Nmap, netdiscover, PCI Dump, Nessus, Wireshark

Skills Needed

Proficiency with network protocols, architecture mapping, and common ports/services expected.

Expected Output

Documentation of assets, network topology, data flows, software/firmware versions, open ports and services.

FDA Mapping

Supports robust testing, attack surface analysis and vulnerability chaining.

Limitations

Sophisticated adversaries can evade detection. Compensating controls should be considered for stealthy reconnaissance tactics.

Vulnerability Analysis

Vulnerability analysis aims to systematically discover flaws and weaknesses in the medical device ecosystem that could be exploited by an attacker to impact safety or effectiveness. This involves a combination of automated scanning, manual verification, and source code and binary review. Key vulnerability analysis activities include:

Automated Scanning - Leverage vulnerability scanners that check for known vulnerabilities based on up-to-date signatures, misconfigurations, and common exposures.

Manual Verification - Skilled testers manually confirm all findings from automated scans to eliminate false positives and further investigate potential issues.

Source Code Review - Manual code reviews and static/dynamic analysis checks to identify risky coding practices, input validation issues, memory management flaws.

Binary Analysis - Analyze firmware images for insecure functions, hardcoded credentials, unpack known vulnerabilities in third-party libraries.

Protocol Analysis - Inspect protocol implementations for weakness related to authentication, encryption, replay prevention, stability.

Tool Examples

Nessus, OpenVAS, Nmap scripts, Burp Suite, Binwalk, Firmadyne, IDA Pro, Wireshark

Skills Needed

Proficiency in porting scanning tools to embedded systems, expertise in common vulnerability categories, secure coding practices, and protocol weaknesses.

Expected Output
Findings categorized by severity, risk vector, conditions required for exploitation including access requirements and chaining prerequisites.

FDA Mapping
Enables robustness analysis, vulnerability chaining, and testing software binaries per FDA guidance.

Limitations
Heavily reliant on signature quality of vulnerability scanning tools, manual verification, and tester skills. Custom and zero-day vulnerabilities can evade detection.

References
For more information, see the following publications: NIST Special Publication 800-115: Technical Guide to Information Security Testing and Assessment, OWASP Vulnerability Scanning Tools, MITRE Common Vulnerabilities and Exposures (CVE), and Penetration Testing Execution Standard Guide (PTES-G).

Authentication Testing

Authentication testing evaluates the strength and proper implementation of mechanisms that verify the identity of users, devices, or services. The goal is to ensure that only authorized entities can access sensitive functionality and data. Key authentication testing techniques include:

Credential Guessing - Attempt to brute force or guess weak, default, or hardcoded passwords.

Session Management - Test for proper invalidation of sessions, secure handling of tokens, and resistance to hijacking.

Multi-Factor Bypass - Attempt to circumvent additional authentication factors like OTP or biometric checks.

Replay Attacks - Capture and resend authentication messages to verify resistance to duplication.

Tools Examples
Hydra, Burp Suite, custom MITM proxies, protocol fuzzers

Skills Needed
Familiarity with authentication protocols, session management best practices, and common authentication bypasses.

Expected Output
Evidence of successful authentication bypass, ability to use weak credentials, or flaws in multi-factor flows leading to unauthorized access.

FDA Mapping
Addresses guidance around strong cryptographic authentication, appropriate access control, updatability authorization, and telemetry integrity verification.

Limitations
Testing may be limited by anti-automation controls on authentication endpoints. Subtle flaws in proprietary authentication schemes can be missed.

References
For more information, see the following publications: NIST Special Publication 800-63B: Digital Identity Guidelines - Authentication and Lifecycle Management, OWASP Authentication Cheat Sheet, and FIDO Alliance Authentication Standards.

Dynamic Analysis Testing

Dynamic analysis testing evaluates device software code in execution to identify vulnerabilities and deviations from secure coding best practices. The goal is confirming flaws manifest at runtime. Techniques include:

Instrumentation - Insert tracking code to monitor variable values, memory operations, function calls to detect overflow or logic errors.

Fuzzing - Provide malformed data as input to cause crashes indicating input validation gaps.

Behavior Modeling - Build patterns of expected execution to detect anomalous control flows.

Tool Examples
Valgrind, AFL, immunity debugger, GDB, proprietary ECU emulation platforms

Skills Required
Secure coding expertise, familiarity with common software flaws, proficiency in advanced debugging interfaces.

Expected Output
Crashes, hangs, unexpected restarts revealing input validation gaps and memory safety errors permitting exploits.

FDA Mapping
Supports guidance areas on software robustness analysis, vulnerability detection, and attack surface mapping.

Limitations
Cannot guarantee all flaws trigger. Limited without source code access. Dependent on test cases and environment model fidelity.

References
For more information, see the following publications: NIST Special Publication 800-53 Revision 5: Security and Privacy Controls for Information Systems and Organizations, OWASP Testing Guide - Dynamic Analysis, and MITRE ATT&CK Framework - Software.

Static Analysis Testing

Static analysis testing examines device software source code and binary executables to identify vulnerabilities and deviations from secure coding best practices without execution. The goal is efficiently augmenting manual reviews. Techniques include:

Flow Analysis - Trace data and control flow to detect buffer overflow potentials.

Symbolic Execution - Determine reachable code paths and feasibility of exploits.

Composition Analysis - Identify included third-party components and derivative works.

Tool Examples
Coverity, Synopsys, Veracode, proprietary medical device static analysis platforms

Skills Required
Secure coding expertise, familiarity with languages and frameworks, understanding common exploit techniques.

Expected Output
Code flaws, architecture weaknesses, risky coding practices requiring security review.

FDA Mapping
Supports guidance areas on software robustness, vulnerability detection, supply chain visibility.

Limitations
Prone to false positives. Limited without source code. Focused on coding mistakes not logical flaws.

References
For more information, see the following publications: NIST Special Publication 500-268: Source Code Security Analysis Tool Functional Specification Version 1.1, OWASP Code Review Guide, and MITRE Common Weakness Enumeration (CWE).

Firmware and Software Reverse Engineering

Reverse engineering thoroughly analyzes device firmware, software, and binaries to identify vulnerabilities, extract hardcoded secrets, and assess the security posture. Techniques include:

Firmware Extraction - Obtain firmware images from hardware interfaces or device storage.

Disassembly/Decompilation - Transform binary code into intermediate representations amenable to analysis.

Debugging - Dynamically step through execution while instrumenting code and memory.

Security Material Discovery - Analyze firmware for security-related data, such as credentials, key

material, randomness generation, and cryptographic seeds.

Behavioral Analysis - Monitor software execution to infer functionality from responses.

Code Review - Manual examination of source code or disassembly for flaws.

Tool Examples
IDA Pro, Ghidra, JEB3, angr, hardware protocol analyzers, debugger hardware

Skills Needed
Experience reversing compiler optimizations, firmware internals, ability to read assembly/bytecode.

Expected Output
Source code, proprietary algorithms, secrets extraction, vulnerability discovery through manual analysis.

FDA Mapping
Supports guidance areas on vulnerability identification, integrity of software binaries, and risks from hardcoded secrets.

Limitations
Time intensive, relies on specialist skills, risks keeping exploiting knowledge.

References
For more information, see the following publications: OWASP Embedded Application Security Project, NIST Special Publication 800-193: Platform Firmware Resiliency Guidelines, and Ghidra, NSA's software reverse engineering tool.

Cloud, Web, and API Testing

Cloud system, web application and API testing evaluates device-facing software services for security issues like injection attacks, improper access control, and lack of encryption. Techniques include:

Fuzzing - Submit malformed, unexpected requests and assess handling.

Authentication - Test account privileges, attempt horizontal/vertical movement between roles.

Session Management - Check for leakage of tokens/cookies, protections on timeouts.

Access Control - Verify consistency of request rejection, confirm restrictions aren't advisory.

Cryptography - Inspect protocol selection, cipher choices, visibility of keys/secrets.

Tool Examples
Burp Suite, OWASP ZAP, Nessus Web Application Scanning

Skills Required
Expertise in common web vulnerabilities, secure API constructs, and testing methodologies like OWASP guidelines.

Expected Output
Vulnerabilities permitting unauthorized data access, operational control, denial of service.

FDA Mapping
Validates guidance recommendations around access control, encryption, and secure handling of authentication and sessions.

Limitations
Effectiveness limited by authentication bound constraints. Custom services can bypass common tests.

References
For more information, see the following publications: OWASP Web Security Testing Guide, NIST Special Publication 800-95: Guide to Secure Web Services, and Cloud Security Alliance Medical Device Risk Management Guide.

Mobile Application Testing

Mobile application testing assesses the security of apps running on mobile devices that interface with medical devices for control or monitoring. Techniques include:

Authentication - Attempt brute force, bypass, and leakage of credentials to impersonate users.

Encryption - Confirm transport security mechanisms are used and properly implemented.

Code Tampering - Manipulate execution through runtime instrumentation to bypass checks.

Data Storage - Check protections and encryption of files persisted locally.

Platform Interaction - Verify app capabilities are appropriately sandboxed from the system.

Tool Examples
objection, MobSF, commercial dynamic and static analyzers designed for mobile

Skills Required
Mobile operating systems internals expertise for iOS and Android, securing coding practices for apps, familiarity with compatibility layers.

Expected Output
Evidence of insecure data handling, ability to manipulate app behavior or access unauthorized functions.

FDA Mapping
Supports guidance areas on data storage protections, appropriate authentication, and secure communication with devices.

Limitations
Heavily dependent on source code availability. Proprietary APIs and frameworks restrict testing.

References
For more information, see the following publications: OWASP Mobile Application Security Verification Standard (MASVS), NIST Special Publication 800-163: Vetting the Security of Mobile Applications, and ENISA Smartphone Secure Development Guidelines.

Fuzz Testing

Fuzz testing involves programmatically manipulating inputs to interfaces and protocol parsers to induce failures that indicate vulnerabilities exist. The goal is to automate exploration of possibility spaces that are infeasible manually. Fuzz testing is deployed as a part of other methods, such as Web Application and API Testing, Mobile Application Testing, Authentication Testing, and Vulnerability Analysis. Key Fuzz Testing activities include:

Corpus Development - Prepare valid and semantic content covering message types and parameter combinations for the interface.

Mutation Engines - Recursively apply transformations like removing elements, changing sizes, injecting unexpected data.

Monitoring - Instrument components under test to capture exceptions and behavioral anomalies.

Prioritization - Focus mutations on critical variables like length fields based on code structure.

Result Analysis - Pinpoint input triggers that disrupt control flow or compromise data integrity from crashes.

Tool Examples
SPIKE, Peach Fuzzer, Sulley, Radamsa, BooFuzz

Skills Required
Familiarity with common software flaws triggered through fuzzing like buffer overflows. Expertise in developing mutation rules tailored to target.

Expected Output
Crashes and unexpected behaviors that reveal input validation gaps permitting memory corruption or unexpected state changes.

FDA Mapping
Supports guidance areas on software robustness analysis and vulnerability detection.

Limitations
Cannot directly prove exploitability absent additional test steps. Skill to develop smart mutation generators and corpora.

References
For more information, see the following publications: OWASP Fuzzing, NIST Special Publication 800-125B: Secure Virtual Network Configuration for Virtual Machine (VM) Protection, and American Fuzzy Lop (AFL) Fuzzer.

Vulnerability Chaining

Vulnerability chaining aims to identify methods to achieve higher impact effects by linking multiple vulnerabilities, sometimes across systems and components.This demonstrates the need for resilience protections and in-depth defense. The goal is confirming potential chains and quantifying their effects. Key vulnerability chaining activities include:

Pivot Analysis - Thoroughly analyze prior findings for prerequisites and incremental gains in access or privileges.

Chain Modeling - Methodically map out pathways connecting vulnerabilities towards asset compromise or patient harm.

Orchestration - Coordinate triggering multiple vulnerabilities, ensuring proper sequence and timing to prevent breakage.

Impact Validation - Demonstrate full chain execution to prove reachability of end goals like data theft or operational disruption.

Parameter Manipulation - Adjust factors like user roles, system configurations, and network rules to bypass roadblocks.

Tool Examples

Custom attack scripts and software that builds off vulnerability analysis findings and automates key steps.

Skills Needed

Methodical skills for combining vulnerabilities, programming for automating key aspects of chains.

Expected Output

Working proof-of-concepts that demonstrate chaining feasibility, quantify incremental gains towards goals.

FDA Mapping

Informs guidance on vulnerability chaining, attack surface depth, robustness.

Limitations

Time-intensive manual analysis and custom software. Defensive controls can introduce roadblocks.

References

For more information, see the following publications: MITRE ATT&CK Framework, PTES Technical Guidelines - Post Exploitation, and OWASP IoT Attack Surface Areas Project.

Exploitation Testing

Exploitation testing entails leveraging vulnerabilities and weaknesses discovered during analysis to demonstrate real-world compromise scenarios. The goal is experimentally confirming flaws are exploitable and quantifying system impact. Key exploitation testing activities include:

Automated Exploits - Leverage frameworks like Metasploit that codify and automate known exploit techniques to achieve code execution, data access, denial of service.

Manual Testing - Skilled testers manually step through chaining discrete vulnerabilities to achieve escalation, such as using XSS to extract session tokens to takeover accounts.

Custom Software - Develop purpose-built software to trigger zero-day and advanced exploits, like fuzzing file parsers to achieve remote code execution.

Hardware Tampering - Test physical hardware interfaces through debugging ports, storage media, protocol signaling manipulation to extract firmware and data.

Tool Examples

Metasploit, attack proxies (e.g. Burp Suite), network manipulation tools, firmware emulators, custom software

Skills Needed

Familiarity with types of vulnerabilities, experience chaining multiple vectors, programming for custom exploits like memory corruption, file format parsing.

Expected Output
Proof of vulnerabilities being exploitable to violate security properties given prerequisites. Severity validated experimentally accounting for defenses.

FDA Mapping
Demonstrates robustness, attack surface depth, supports vulnerability chaining analysis.

Limitations
Ethical constraints, restricted access, and compensating controls may limit exploitation severity. Custom exploits require advanced skills.

References
For more information, see the following publications: PTES Technical Guidelines - Exploitation, OWASP Testing Guide - Testing for Weak Cryptography, and Metasploit Framework Documentation.

Kiosk Escape and Touch Interface Testing

Evaluate the degree to which the application, operating system, ports, or account settings allowed for an escape of the primary user interface. This includes verification of kiosk mode, how it is enabled, and attempts to access the underlying operating system through manual and automated tools/techniques. Common approaches include:

Touchscreen Tampering - Manipulate sensitivity or trigger touch events outside expected regions.

Browser Manipulation - Force new tabs/windows outside the intended application container through JavaScript injection or shortcut exploits.

Window Manager Attacks - Manipulate window focus and visibility restrictions imposed by kiosk mode managers.

Sandbox Bypasses - Leverage application logic flaws or wrapper mismatches to allow execution outside intended data and network constraints.

Authentication Compromise - Steal credentials or session tokens using clipboard logging, keystroke sniffing, or credential proxying to impersonate valid users.

Privilege Escalation - Chain application vulnerabilities to pivot from low privilege clinical software roles to achieve administrator system rights.

UEFI/BIOS Tampering - Attempting to access the UEFI or BIOS menu to tamper with settings.

External Boot - Starting the system or a component from an unauthorized device, such as a USB drive.

Tools Examples
Burp Suite, Frida, custom phishing proxies, credential harvesting tools

Skills Required
Expertise in software vulnerabilities, JavaScript, familiarity with medical systems workflows, privilege escalation techniques

Expected Output
Evidence of unconstrained system access, ability to reach other internal resources, access captured from stolen credentials

FDA Mapping
Informs guidance on integrity protections, demonstrating authentication control weaknesses.

Limitations
Restricted utility within physical tester access constraints. Dependent on application logic flaws risks are overstated.

References
For more information, see the following documentation: NIST Special Publication 800-124 Revision 2: Guidelines for Managing the Security of Mobile Devices in the Enterprise, CIS Benchmarks - Kiosk Lockdown, and Cracking the Code: Escaping Kiosk Mode and Unraveling Dispenser Board Firmware Insecurity.

Enclosure Tamper Testing

Enclosure tampering testing attempts to gain unauthorized internal access to device hardware and electronics by circumventing physical chassis protections. This demonstrates the sufficiency of

tamper resistance and evidence mechanisms. Approaches include:

Tamper Evident Bypass - Circumventing adhesive seals or fragile stickers indicating previous opening through steam, solvents, or carefully working edges.

Destructive Entry - Breaching device enclosures through cutting, drilling, or snapping plastic welds when stealth is not required. This is done on a limited basis and only after discussion with the client - typically only on consumables.

Security Screw Defeat - Using commercial bits, nails, or melted polymer to remove specialty screw heads designed to bind standard drivers.

Depotting - Removing protective resin or epoxy covering electronics using chemical means, heat, or mechanical force. Enables access while avoiding damage to the electronics.

Tool Examples
Plastic wedges, precision screwdrivers, lockpicks, epoxy strippers, drills, proprietary security bits

Skills Required
Hardware tampering techniques, familiarity with tamper evidence, medical device internals knowledge

Expected Output
Successful access to restricted internal hardware provides a blueprint for intellectual property theft or operational manipulation in succeeding steps.

FDA Mapping
Informs guidance on ensuring authenticity and integrity of software binaries by defeating hardware protections.

Limitations
Invasive testing risks equipment damage. Dependent on physical access which can be constrained. Limited evidence of exploitability.

References
For more information, see the following publications: NIST Special Publication 800-88 Revision 1: Guidelines for Media Sanitization, OWASP Embedded Application Security Best Practices, Chip Trimming and Decapping, and JTAG (Joint Test Action Group) IEEE 1149.1 Standard.

Hardware Layer Evaluation

Physical controls bypass testing attempts to circumvent hardware-based security mechanisms through standard interfaces or physical access manipulation. The goal is confirming sufficiency of protections against invasive attacks. Key physical controls bypass activities include:

Interface Level Access - Use specialized tools like JTAG debuggers, UART console cables, or CAN bus adapters to interact with chipsets directly.

Firmware Extraction - Leverage interface access to retrieve memory contents for analyzing security defenses.

Storage Scrubbing - Attempt to recover supposedly scrubbed sensitive data from flash storage chips through commercial tools.

Glitching - Discover data in memory, such as keys, firmware, or protected data by abruptly cutting then restoring power during operation.

Tool Examples
Bus Pirate, GPIO cables, JTAG debugging tools, lock picking tools, hardware protocol analyzers

Skills Needed
Hardware tampering techniques, embedded system internals knowledge, physical security controls familiarity.

Expected Output
Evidence of extracted firmware, console access achieved, ability to bypass authentication controls through hardware interfaces.

FDA Mapping
Verifying hardware protections around authentication, software binaries, integrity assurance per guidance areas.

Limitations
Invasive testing has equipment damage risks requiring fail-safes. Strictly controlled environments may impede manipulation access.