Best Practices for API Compliance & Privacy

Dan Barahona
dan@apisec.ai
apisec.ai

Hackers have taken notice of the prevalence of APIs, and the existence of serious vulnerabilities within these interfaces. Daily, new API-based breaches occur with hackers taking advantage of undiscovered loopholes in API logic, allowing malicious access to sensitive data. This paper examines the compliance and privacy challenges and best practices for securing APIs.

Privacy

APIs drive nearly every function online - utilized to enable everything from sending money from a mobile app to syncing health data across providers to powering social communities, dating sites, and credit reporting agencies.

As a result, APIs now have access to the most sensitive data and functionality within organizations. And as we've seen in many, many public breaches, these APIs are often under-secured and over-permissioned, becoming the source of significant breaches and resulting in the loss of hundreds of millions of records. Here are a few examples:

In 2018 the US Postal Service offered an API called "Informed Visibility" that allowed businesses to access information about packages, shipments, and mailings in real time. While the API required users to be authorized to the service and to authenticate to use it, there was a flaw in the API's logic. A request to the API could be manipulated to request information of another user, which the server would deliver, providing access to over 60 million users' information, including email address, account number, street address, phone number and more.

In 2019, the money transfer site, Venmo, had over 200 million transactions harvested, including sensitive user information. The hacker discovered an unsecured API endpoint that was used by the company homepage to present recent transactions. The homepage removed any sensitive information. However, the API, when called directly, returned all the transaction details, and because the endpoint was unsecured, there was no authentication required.

In 2020, the dating site Bumble left an open API endpoint without authentication, allowing access to nearly 100 million users' information. Of note, the API would also return the physical distance to another member. By setting up three accounts with different locations the hacker could identify the exact location of any other user.

These examples highlight the potential for large scale privacy loss via APIs. It's not surprising Gartner predicts APIs will become the "most frequent attack vector" by 2022. Therefore, the implications of APIs regarding privacy are huge, as are the potential financial damages from fines and lawsuits. Privacy regulations such as GDPR in Europe, CCPA in California, HIPAA for health privacy, mandate the protection of an individual's private information and set significant penalties for loss. Gartner reports that over 60 countries have enacted privacy mandates around the world.

GDPR

The European Union passed the "toughest privacy and security law in the world," enacting the General Data Protection Regulation (GDPR) in 2018. The law sets strict obligations for the protection and control of privacy information, applying to any provider that handles any such information. Further, GDPR imposes severe fines for violations, up to 4% of total global revenue or 20 million Euros, whichever is higher.

GDPR defines 7 specific requirements for the control of personally identifiable information (PII) - item #6 covers Integrity and Confidentiality, requiring that personal data is "processed in a manner that ensures appropriate security of personal data, including protection against unauthorized or unlawful processing and accidental loss, destruction, or damage, using appropriate technical or organizational measures." The Data Security requirements mandate "appropriate technical and organizational measures."

While GDPR does not specify the exact measures to be taken to secure PII, it does mandate the protection of this information and sets significant penalties for any failures or breaches. Below we provide details on how organizations need to approach APIs to maintain GDPR compliance.

CCPA

In California, the State Legislature enacted a privacy regulation similar to GDPR, becoming enforced no later than January 1, 2020. The California Consumer Privacy Act

(CCPA) sets requirements on organizations that handle PII to control what data is collected, how it's used and secured, and the rights of individuals to control and revoke their information. Unlike GDPR, CCPA does not mandate a specific user opt-in to collect this information.

CCPA does not mandate any technical controls for data security, or any requirements for APIs specifically. CCPA focuses on the core objective of ensuring PII is handled securely. It also sets penalties for violations - ranging from $100 to $750 per violation. With breaches often measured in millions of records, the penalties can be massive. CCPA also provides a "private right of action that allows consumers to sue businesses when their non-encrypted or non-redacted personal information is subject to unauthorized access. This includes exfiltration, theft, or disclosing due to the business' violation of its duty to implement and maintain reasonable security procedures and practices appropriate to the nature of the information."

HIPAA

The Health Information Portability and Accountability Act of 1996 (HIPAA) was likely the first major privacy legislation, applying specifically to the protection of health-related personal information. The regulation includes both a HIPAA Privacy Rule and HIPAA Security Rule. The Privacy Rule defines what organizations are covered, what information is protected and rules for permitted use and disclosure. The rule also sets limitations on how personal medical information can be used and defines penalties for violations, from $100 to $50,000 per violations. HIPAA also imposes criminal penalties for knowingly obtaining protected information - up to one year in prison.

The Security Rule sets requirements for the protection of PII, including ensuring "the confidentiality, integrity, and availability of all e-PHI [organizations] create, receive, maintain or transmit." The rule also mandates that organizations "Protect against reasonably anticipated, impermissible uses or disclosures," and that "covered entities to perform risk analysis as part of their security management processes." This risk

assessment must address any technical mechanisms in which PII can be accessed or manipulated, and clearly encompasses any APIs that touch PII.

HIPAA goes further than GDPR and CCPA to mandate certain technical controls, albeit still at a high level. For example, HIPAA requires that “A covered entity must implement technical policies and procedures that allow only authorized persons to access electronic protected health information (e-PHI).” Virtually online medical record systems leverage APIs, especially when sharing information across organizations - these APIs must be highly secured and validated to ensure no unauthorized access is possible.

Privacy Implications for APIs

Across all these privacy regulations we see a focus on the objective of protecting personal information, while leaving out specifics on how to protect this data. CCPA specifies that organizations must have “reasonable security,” whereas GDPR requires “appropriate security” measures be put in place. No specific direction is provided to dictate how organizations must secure this sensitive information.

This puts the responsibility on organizations to perform a thorough risk assessment, understand the mechanisms available to access PII, and implement appropriate tests and controls to ensure the data is kept secure. Suppose APIs are being used to access and transmit personal information. In that case, these APIs need to be part of risk assessments, and organizations must determine what controls are appropriate, and explain how risks and vulnerabilities will be identified and addressed. Consent is also required because if data is being moved around in ways that you’re not fully aware of via APIs, then you don’t have the appropriate consent for that movement. While technical controls, such as authentication, authorization and monitoring are needed, it is also critical to perform regular and comprehensive security tests to uncover any gaps in controls and unintended logic faults that can give a hacker access to protected information.

Compliance

Security regulations have existed now for decades, with the introduction of HIPAA in healthcare, PCI in credit cards, GLBA in banking, NERC in energy, and many more. These regulations emerged to address cybersecurity threats and establish minimum levels of security to ensure smooth, reliable and trustworthy operations within various industries. These regulations have long addressed the risk of breach to applications and infrastructure that can lead to service disruption, data theft, and other exploits.

As APIs have become pervasive across all these sectors, and more, they fall directly under these regulatory regimes and organizations must factor in APIs into their compliance plans. APIs are often made available publicly, providing an easy attack vector that allows access to sensitive customer information and internal data.

Below we explore compliance requirements across multiple industries - however, this is not meant to be an exhaustive list. The key point is API security is absolutely relevant to any cybersecurity regulation that mandates the protection of data and applications.

PCI DSS

The Payment Card Industry Data Security Standard (PCI DSS) was created to ensure secure and trusted electronic payments and transactions. The regulation is much more explicit than the privacy regulations about what capabilities are mandated for payment processors. PCI DSS is comprised of 12 requirements covering everything from the use of firewalls to performing regular vulnerability scans to implementing access controls.

The following specific sections of the PCI DSS directly apply to APIs. Like any web, mobile, or internal application, APIs are subject to vulnerabilities and these must be discovered and remediated as early as possible, before code is pushed to production.

- **Section 6.1:** Establish a process to identify security vulnerabilities
- **Section 6.3:** Develop software applications based on industry best practices and incorporate information security throughout the software development life cycle
- **Section 6.5:** Develop all Web applications based on secure coding guidelines and review custom application code to identify coding vulnerabilities
- **Section 6.6:** For public-facing web applications, address new threats and vulnerabilities on an ongoing basis and ensure these applications are protected against known attacks
- **Section 11.3:** Perform external and internal penetration testing at least once a year and after any significant application upgrade or modification, including application-layer penetration tests

The PCI DSS is very explicit about the need to test applications to find any vulnerabilities, including after any application modifications. This is especially important for APIs as this organizations commonly push updates monthly, weekly and even daily. While a full manual penetration test is generally impractical, new API testing solutions exist that can automate this and provide continuous API security assessment.

FedRAMP

The US Federal government created the Federal Risk and Authorization Management Program (FedRAMP) to provide a standardized approach to certifying cloud-based services for use in the US Government. Security is at the core of FedRAMP, which provides very explicit and detailed security requirements for any Cloud Service Provider (CSP) to be authorized for government use. These requirements encompass topics including continuous monitoring, security controls

The FedRAMP Program Management Office (PMO) issued a set of guidelines specifically to address the need to search for and uncover any vulnerabilities in applications and infrastructure. The regulation mandates that “CSPs must scan web applications monthly” and goes on to clarify that scans must cover “All web interfaces

and services." This language specifically includes all forms of web-facing interface, including application programming interfaces.

Open Banking

The European Union enacted in 2016 the Payment Services Directive (PSD2) to enable more seamless financial interoperability across institutions. PSD2 led to the creation of an Open Banking standard for all institutions to support. At the core of Open Banking is a set of Banking APIs, developed to standardize the movement of funds, transactions and information between organizations.

APIs form the backbone of Open Banking and their security is absolutely critical. As seen in many publicized breaches, APIs often contain loopholes, excess permissions and logic flaws that can lead to significant breaches. The PSD2 specifically recommends financial institutions adopt the ISO 27001 cybersecurity framework to address security needs.

Open Banking presents a unique challenge to covered institutions - on one hand mandating the use of APIs and requiring 3rd party interoperability, and on the other insisting that all transfers and transactions are kept absolutely secure and private.

Healthcare Interoperability

The Centers for Medicaid and Medicare Services (CMS) has set out to improve the access to health information for patients and providers, and to enable digital interoperability to streamline data exchange. CMS created the Interoperability and Patient Access Rule to set requirements and standards for the secure and efficient sharing of health information. At the core of the Interoperability Rule is a suite of defined APIs leveraging the Health Level 7 (HL7) Fast Healthcare Interoperability Resources (FHIR). The Interoperability mandate became effective January 1, 2021.

As with the Open Banking requirements, healthcare providers are faced with dueling mandates - to ensure patient privacy and data protection (HIPAA) and to enable

seamless, electronic data access across organizations (CMS Interoperability). To satisfy these competing requirements healthcare organizations must employ APIs and ensure that those APIs are highly secure. Again, the regulation does not specify how to achieve security, relying on industry best practices and standard frameworks to provide this direction. But with APIs at the center of the Interoperability guideline, organizations must test all APIs, continuously, and with complete coverage to ensure no logic flaws or security vulnerabilities can be exploited to access PII.

Compliance Implications for APIs

As we see here, cybersecurity regulations exist in virtually every major industry, and generally share the same core objectives to ensure the reliable, secure, and trusted operation of critical services. These regulations differ significantly in how prescriptive they are in specifying how covered organizations must achieve the objective. PCI and FedRAMP are very clear about what types of capabilities, controls, and mechanisms, including the need to focus on discovery of vulnerabilities at the application and API level.

Other regulations, such as the Open Banking and CMS Interoperability Rule, leave it to each regulated organization to figure out how best to comply. But even here the expectation is that companies will follow industry best practices, employ standard security frameworks like SOC 2 and ISO 27001, and perform a comprehensive risk assessment. Wherever APIs exist, auditors will be looking for appropriate controls and measures.

Approaches to API Security

The OWASP organization recognized the fundamental differences in security risk for APIs versus web and mobile apps. Similarly, the widely used tools and techniques used to secure web and mobile apps do not offer the same protection for APIs. New approaches are required for securing APIs - these approaches generally fall into three categories:

1. Static Code Analysis: building secure APIs during development
2. Security Testing: security and vulnerability testing
3. Application Firewall: protecting live APIs with inline/traffic solutions

Building security testing into a company's CI/CD pipeline is a common first line of defense. This approach focuses on the code itself, adding Static Analysis Security Testing (SAST) and other code quality solutions to the continuous integration workflow. There are well established vendors in this segment, and SAST can help enormously with ensuring the quality and maintainability of a codebase and to identify common, well-known coding issues and vulnerabilities. However, static code analysis is incapable of identifying the types of logic flaws that lead to major API breaches and data loss.

Operations often deploy a second line of defense by deploying Web Application Firewalls (WAF) and API-aware traffic inspectors to the production API environments. These firewalls analyze network traffic and employ heuristic techniques to watch for common attack patterns. API-aware firewalls can go a step further, looking for API-specific anomalies, such as preventing strings from being passed to an API that should only receive integers. This is a capability that would be useful across all APIs.

However, such technologies would not generally be able to identify an API user that is attempting to access data belonging to another user. This requires a level of user visibility and session awareness that API-aware firewalls lack, despite attribution techniques including IP addresses, packet headers, behavioral patterns, and user-agents that are employed in web application firewalls.

The primary issues with code analysis and API firewalling is that these technologies focus primarily on classic security-oriented vulnerabilities such as SQL injection attacks, cross-site scripting, buffer overflows, etc. As we saw earlier, these are not the causes of the vast majority of API breaches - these breaches are a result of the business logic flaws that are unique to each API and cannot be detected or prevented with standard approaches.

Lastly, companies invest in manual penetration testing by security professionals or internal Red Teams. Pen-testing relies on humans to understand API code, craft tests to identify vulnerabilities, execute the tests, and then interpret the results. This manual approach is, by definition, slow (weeks, not minutes), reactive (often a once or twice a year effort), and costly. Security vulnerabilities are only discovered after they have reached production. Despite these drawbacks, manual pen testing approaches security from the mindset of an adversarial attacker, adding depth in defending against security attacks.

Meanwhile, organizations continue to operate at the speed of DevOps, with new code pushed to production every day. As a result, defects make it into production, creating vulnerabilities and leading to serious breaches as described earlier.

Challenges of API Security Testing

With the move towards API-driven applications and functionality, the challenge of security testing code gets even harder. There are three main issues: coverage, scale, and speed.

Coverage

The ultimate goal of security testing is to make code less vulnerable to attack, misuse and exploits. To achieve this testing must cover every corner of the API's capabilities, not just what is expected. Corey Ball, author of the upcoming book "Hacking APIs" puts it succinctly, "You can design an API you think is ultra-secure, but if you don't test it, then a cybercriminal somewhere is going to do it for you." So the testing scheme needs to look at every API endpoint and method, and consider all the possible ways your API can be used, not just the likely ways.

Scale

The coverage problem leads to the scale problem. How do you create test scenarios to cover the entire range of API functionality. Consider a relatively lightweight API with, say, 50 endpoints. Each of those endpoints can support multiple POST, GET, PUT and DELETE methods. Quickly you're already up to ~250 endpoint-method permutations. And then consider the myriad API breach categories as described by the OWASP API Security Top 10 - pushing testing requirements to thousands of unique attacks. Testing needs to cover all these permutations and scenarios, otherwise vulnerabilities will make it to production waiting to get discovered by someone else.

Speed

Speed in CI/CD time means seconds or minutes. When fixes and new functionality need to move to production, testing cannot delay progress. However, given the complexity of APIs, the breadth of scenarios to cover, testing speed is more often measured in weeks or months.

A Better Way: Automated API Security Testing

The coverage problem is multiplied by the scale problem - how to create test scenarios to cover the entire range of API functionality. Consider a relatively lightweight API with, say, 50 endpoints. Each of those endpoints can support multiple POST, GET, PUT and DELETE methods. Quickly you're already up to 200+ endpoint-method permutations. And then consider the myriad API breach categories

The APIsec approach to API security begins with learning the API: cataloging all available endpoints and identifying supported methods. This approach is 100% automated and allows critical API vulnerabilities to be addressed before a company's product reaches Production. This allows approach offers major advantages to static code analysis, API firewalling, and manual pen testing:

1. **Critical Vulnerability Detection** - discovers vulnerabilities missed by static analysis and firewalls, including business logic faults and access control issues.

2. **Automatic Test Creation** - automatically creates thousands of test sequences with no manual effort. This frees a team from the necessity of dreaming up every possible test scenario and hand crafting tests, potentially saving thousands of hours of effort.

3. **Complete API Coverage** – creates granular tests to cover a company's entire API footprint, addressing all API attack breach categories. It covers all of the nooks, crannies, edge cases, and corners that are easy to miss with manual efforts.

4. **Speed of DevSecOps** – enables API testing on every nightly build or release, without adding any delay. Testing is fast, running in minutes rather than adding hours or days to a company's integration and deployment workflow.

5. **Continuous Security** – provides continuous API protection, unlike manual pen-testing that is run monthly or at longer intervals.

6. **Cost efficiency** – provides far greater test coverage of manual pen-testing at much lower cost, leveraging the efficiencies inherent in a scalable and automated process.

After completing the vulnerability analysis, APIsec integrates results into the CI/CD pipeline, creates trouble tickets in popular systems like Github and Jira, produces pen-test reports suitable for submission to regulatory auditors and compliance officers, and provides an easy-to-use dashboard for visualizing and managing changes to an API over time. The dashboard allows replaying the attack to show the exact nature of the vulnerability, and automatically calculates Common Vulnerability Scoring System (CVSS) severity scores.

Please visit www.apisec.ai to learn more.

Dan Barahona, Head of Business Development
Email: dan@apisec.ai