Smarttech
YOUR 24/7 SECURITY PARTNER

Multiple Vulnerabilities in OpenSSL

Document ID	SMA-Threat Report
Document status	ISSUED
Issue Number	15
Authors	Dorin Constantin Banu < constantin.banu@smarttech247.com >
Verified by	Alin Curcan < alin.curcan@smarttech247.com >
Last modified	2026-01-29
Issue Date	2026-01-29

Threat Reports are reports created by Smarttech247 based on high and critical severity vulnerabilities that may have a high potential to be exploited in the wild i.e. vulnerabilities that are present in most used products by companies and do not have an auto-update option or they are usually not automatically updated in case that could lead to some service disruption. This report is usually created as soon as the vulnerability is released, therefore we strongly recommend that the information is reviewed, tests are performed and patches are applied before the first proof-of-concept is released. Even though certain vulnerabilities may not have an active exploit in the wild at the time that we report on them, we take into consideration the wider risk and the impact it could have on systems, should an exploit like that be available after a while. Our duty is to report them on time and we recommend enterprises that, in order to keep critical business systems protected, they should consider, on average, ten working days to check whether or not the new vulnerability affects them, and if so, to implement actions in order to remove the risk.

Overview:

Multiple vulnerabilities have been discovered in OpenSSL, which could allow for remote code execution, Denial of Service (DoS), integrity bypass, memory corruption and information disclosure. OpenSSL is a software library for applications that provide secure communications over computer networks against eavesdropping and identify the party at the other end. It is widely used by Internet servers, including the majority of HTTPS websites.

Risk

Government:
- Large and medium government entities: High
- Small government entities: High

Businesses:
- Large and medium business entities: High
- Small business entities: High

Technical summary

CVE ID	***Description***	***Impact***
CVE-2025-11187	When verifying a PKCS#12 file that uses PBMAC1 for the MAC, the PBKDF2 salt and keylength parameters from the file are used without validation. If the value of keylength exceeds the size of the fixed stack buffer used for the derived key (64 bytes), the key derivation will overflow the buffer. The overflow length is attacker-controlled. Also, if the salt parameter is not an OCTET STRING type this can lead to invalid or NULL pointer dereference. Exploiting this issue requires a user or application to process a maliciously crafted PKCS#12 file.	Denial of Service (DoS) and code execution
CVE-2025-15467	When parsing CMS AuthEnvelopedData structures that use AEAD ciphers such as AES-GCM, the IV (Initialization Vector) encoded in the ASN.1 parameters is copied into a fixed-size stack buffer without verifying that its length fits the destination. An attacker can supply a crafted CMS message with an oversized IV, causing a stack-based out-of-bounds write before any authentication or tag verification occurs. Applications and services that parse untrusted CMS or PKCS#7 content using AEAD ciphers (e.g., S/MIME AuthEnvelopedData with AES-GCM) are vulnerable. Because the overflow occurs prior to authentication, no valid key material is required to trigger it.	Denial of Service (DoS) and remote code execution

CVE-2025-15468	Some applications call SSL_CIPHER_find() from the client_hello_cb callback on the cipher ID received from the peer. If this is done with an SSL object implementing the QUIC protocol, NULL pointer dereference will happen if the examined cipher ID is unknown or unsupported.	Denial of Service (DoS)
CVE-2025-15469	When the 'openssl dgst' command is used with algorithms that only support one-shot signing (Ed25519, Ed448, ML-DSA-44, ML-DSA-65, ML-DSA-87), the input is buffered with a 16MB limit. If the input exceeds this limit, the tool silently truncates to the first 16MB and continues without signaling an error, contrary to what the documentation states. This creates an integrity gap where trailing bytes can be modified without detection if both signing and verification are performed using the same affected codepath. The issue affects only the command-line tool behavior.	Integrity bypass
CVE-2025-66199	In affected configurations, the peer-supplied uncompressed certificate length from a CompressedCertificate message is used to grow a heap buffer prior to decompression. This length is not bounded by the max_cert_list setting, which otherwise constrains certificate message sizes. An attacker can exploit this to cause large per-connection allocations followed by handshake failure. No memory corruption or information disclosure occurs. This issue only affects builds where TLS 1.3 certificate compression is compiled in (i.e., not OPENSSL_NO_COMP_ALG) and at least one compression algorithm (brotli, zlib, or zstd) is available, and where the compression extension is negotiated. Both clients receiving a server CompressedCertificate and servers in mutual TLS scenarios receiving a client CompressedCertificate are affected. Servers that do not request client certificates are not vulnerable to client-initiated attacks. Users can mitigate this issue by setting SSL_OP_NO_RX_CERTIFICATE_COMPRESSION to disable receiving compressed certificates.	Denial of Service (DoS)
CVE-2025-68160	The line-buffering BIO filter (BIO_f_linebuffer) is not used by default in TLS/SSL data paths. In OpenSSL command-line applications, it is typically only pushed onto stdout/stderr on VMS systems. Third-party applications that explicitly use this filter with a BIO chain that can short-write and that write large, newline-free data influenced by an attacker would be affected. However, the circumstances where this could happen are unlikely to be under attacker control, and BIO_f_linebuffer is unlikely to be handling non-curated data controlled by an attacker.	Denial of Service (DoS)
CVE-2025-69418	The low-level OCB encrypt and decrypt routines in the hardware-accelerated stream path process full 16-byte blocks but do not advance the input/output pointers. The subsequent tail-handling code then operates on the original base pointers, effectively reprocessing the beginning of the buffer while leaving the actual trailing bytes unprocessed. The authentication checksum also excludes the true tail bytes. However, typical OpenSSL consumers using EVP are not affected because the higher-level EVP and provider OCB implementations split inputs so that full blocks and trailing partial blocks are processed in separate calls, avoiding the problematic code path. Additionally, TLS does not use OCB ciphersuites. The vulnerability only affects applications that call the low-level CRYPTO_ocb128_encrypt() or CRYPTO_ocb128_decrypt() functions directly with non-block-aligned lengths in a single call on hardware-accelerated builds.	Information disclosure

CVE-2025-69419	The OPENSSL_uni2utf8() function performs a two-pass conversion of a PKCS#12 BMPString (UTF-16BE) to UTF-8. In the second pass, when emitting UTF-8 bytes, the helper function bmp_to_utf8() incorrectly forwards the remaining UTF-16 source byte count as the destination buffer capacity to UTF8_putc(). For BMP code points above U+07FF, UTF-8 requires three bytes, but the forwarded capacity can be just two bytes. UTF8_putc() then returns -1, and this negative value is added to the output length without validation, causing the length to become negative. The subsequent trailing NUL byte is then written at a negative offset, causing write outside of heap allocated buffer. The vulnerability is reachable via the public PKCS12_get_friendlyname() API when parsing attacker-controlled PKCS#12 files. While PKCS12_parse() uses a different code path that avoids this issue, PKCS12_get_friendlyname() directly invokes the vulnerable function. Exploitation requires an attacker to provide a malicious PKCS#12 file to be parsed by the application and the attacker can just trigger a one zero byte write before the allocated buffer.	Memory corruption and Denial of Service (DoS)
CVE-2025-69420	The functions ossl_ess_get_signing_cert() and ossl_ess_get_signing_cert_v2() access the signing cert attribute value without validating its type. When the type is not V_ASN1_SEQUENCE, this results in accessing invalid memory through the ASN1_TYPE union, causing a crash. Exploiting this vulnerability requires an attacker to provide a malformed TimeStamp Response to an application that verifies timestamp responses.	Denial of Service (DoS)
CVE-2025-69421	The PKCS12_item_decrypt_d2i_ex() function does not check whether the oct parameter is NULL before dereferencing it. When called from PKCS12_unpack_p7encdata() with a malformed PKCS#12 file, this parameter can be NULL, causing a crash. The vulnerability is limited to Denial of Service and cannot be escalated to achieve code execution or memory disclosure. Exploiting this issue requires an attacker to provide a malformed PKCS#12 file to an application that processes it.	Denial of Service (DoS)
CVE-2026-22795	A type confusion vulnerability exists in PKCS#12 parsing code where an ASN1_TYPE union member is accessed without first validating the type, causing an invalid pointer read. The location is constrained to a 1-byte address space, meaning any attempted pointer manipulation can only target addresses between 0x00 and 0xFF. This range corresponds to the zero page, which is unmapped on most modern operating systems and will reliably result in a crash, leading only to a Denial of Service. Exploiting this issue also requires a user or application to process a maliciously crafted PKCS#12 file. It is uncommon to accept untrusted PKCS#12 files in applications as they are usually used to store private keys which are trusted by definition.	Denial of Service (DoS)
CVE-2026-22796	The function PKCS7_digest_from_attributes() accesses the message digest attribute value without validating its type. When the type is not V_ASN1_OCTET_STRING, this results in accessing invalid memory through the ASN1_TYPE union, causing a crash. Exploiting this vulnerability requires an attacker to provide a malformed signed PKCS#7 to an application that verifies it. The impact of the exploit is just a Denial of Service, the PKCS7 API is legacy and applications should be using the CMS API instead.	Denial of Service (DoS)

Affected Versions

Product	Affected	Solution
OpenSSL	from 3.6.0 up to 3.6.1	3.6.1
OpenSSL	from 3.5.0 up to 3.5.5	3.5.5
OpenSSL	from 3.4.0 up to 3.4.4	3.4.4
OpenSSL	from 3.3.0 up to 3.3.6	3.3.6
OpenSSL	from 3.0.0 up to 3.0.19	3.0.19
OpenSSL	from 1.1.1 up to 1.1.1ze	1.1.1ze
OpenSSL	from 1.0.2 up to 1.0.2zn	1.0.2zn

***Note:** All OpenSSL versions before 1.1.1 are out of support and are no longer receiving updates. Extended support is available for 1.0.2 from OpenSSL Software Services for premium support customers.*

Recommendations

Smarttech247 team recommend the following actions be taken:

- Apply appropriate updates provided by OpenSSL to vulnerable systems immediately after appropriate testing. (**M1051**: Update Software)
    - Safeguard 7.1: Establish and Maintain a Vulnerability Management Process: Establish and maintain a documented vulnerability management process for enterprise assets. Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.
    - Safeguard 7.2: Establish and Maintain a Remediation Process: Establish and maintain a risk-based remediation strategy documented in a remediation process, with monthly, or more frequent, reviews.
    - Safeguard 7.4: Perform Automated Application Patch Management: Perform application updates on enterprise assets through automated patch management on a monthly, or more frequent, basis.
    - Safeguard 7.5: Perform Automated Vulnerability Scans of Internal Enterprise Assets: Perform automated vulnerability scans of internal enterprise assets on a quarterly, or more frequent, basis. Conduct both authenticated and unauthenticated scans, using a SCAP-compliant vulnerability scanning tool.
    - Safeguard 7.7: Remediate Detected Vulnerabilities: Remediate detected vulnerabilities in software through processes and tooling on a monthly, or more frequent, basis, based on the remediation process.
    - Safeguard 12.1: Ensure Network Infrastructure is Up-to-Date: Ensure network infrastructure is kept up-to-date. Example implementations include running the latest stable release of software and/or using currently supported network-as-a-service (NaaS) offerings. Review software versions monthly, or more frequently, to verify software support.
    - Safeguard 18.1: Establish and Maintain a Penetration Testing Program: Establish and maintain a penetration testing program appropriate to the size, complexity, and maturity of the enterprise. Penetration testing program characteristics include scope, such as network, web application, Application Programming Interface (API), hosted services, and physical premise controls; frequency; limitations, such as acceptable hours, and excluded attack types; point of contact information; remediation, such as how findings will be routed internally; and retrospective requirements.
    - Safeguard 18.2: Perform Periodic External Penetration Tests: Perform periodic external penetration tests based on program requirements, no less than annually. External penetration testing must include enterprise and environmental reconnaissance to detect exploitable information. Penetration testing requires

specialized skills and experience and must be conducted through a qualified party. The testing may be clear box or opaque box.
  - Safeguard 18.3: Remediate Penetration Test Findings: Remediate penetration test findings based on the enterprise's policy for remediation scope and prioritization.

- Apply the Principle of Least Privilege to all systems and services. Run all software as a non-privileged user (one without administrative privileges) to diminish the effects of a successful attack. (**M1026**: Privileged Account Management)
  - Safeguard 4.7: Manage Default Accounts on Enterprise Assets and Software: Manage default accounts on enterprise assets and software, such as root, administrator, and other pre-configured vendor accounts. Example implementations can include: disabling default accounts or making them unusable.
  - Safeguard 5.5: Establish and Maintain an Inventory of Service Accounts: Establish and maintain an inventory of service accounts. The inventory, at a minimum, must contain department owner, review date, and purpose. Perform service account reviews to validate that all active accounts are authorized, on a recurring schedule at a minimum quarterly, or more frequently.

- Vulnerability scanning is used to find potentially exploitable software vulnerabilities to remediate them. (**M1016**: Vulnerability Scanning)
  - Safeguard 16.13: Conduct Application Penetration Testing: Conduct application penetration testing. For critical applications, authenticated penetration testing is better suited to finding business logic vulnerabilities than code scanning and automated security testing. Penetration testing relies on the skill of the tester to manually manipulate an application as an authenticated and unauthenticated user.

- Architect sections of the network to isolate critical systems, functions, or resources. Use physical and logical segmentation to prevent access to potentially sensitive systems and information. Use a DMZ to contain any internet-facing services that should not be exposed from the internal network. Configure separate virtual private cloud (VPC) instances to isolate critical cloud systems. (**M1030**: Network Segmentation)
  - Safeguard 12.2: Establish and Maintain a Secure Network Architecture: Establish and maintain a secure network architecture. A secure network architecture must address segmentation, least privilege, and availability, at a minimum.

- Use capabilities to detect and block conditions that may lead to or be indicative of a software exploit occurring. (**M1050**: Exploit Protection)
  - Safeguard 10.5: Enable Anti-Exploitation Features: Enable anti-exploitation features on enterprise assets and software, where possible, such as Microsoft® Data Execution Prevention (DEP), Windows® Defender Exploit Guard (WDEG), or Apple® System Integrity Protection (SIP) and Gatekeeper™.

References

https://openssl-library.org/news/vulnerabilities/index.html#CVE-2025-15467
https://www.securityweek.com/high-severity-remote-code-execution-vulnerability-patched-in-openssl/

CVE

CVE-2025-11187
CVE-2025-15467
CVE-2025-15468
CVE-2025-15469

CVE-2025-66199
CVE-2025-68160
CVE-2025-69418
CVE-2025-69419
CVE-2025-69420
CVE-2025-69421
CVE-2026-22795
CVE-2026-22796

Smarttech
YOUR 24/7 SECURITY PARTNER

www.smarttech247.com